
Mesmerize Documentation

Release 0.2.0

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	Add a Sample to the Project	57
9	Tiff file module	59
10	Batch Manager	63
11	Stimulus Mapping	67
12	ROI Manager	71
13	Caiman Motion Correction	75
14	CNMF	83
15	CNMF 3D	87
16	CNMF-E	91
17	Caiman HDF5 Importer	97
18	Suite2p Importer	99
19	Script Editor	101
20	Flowchart Overview	103
21	Nodes	109
22	Examples	131

23 Beeswarm	135
24 Consoles	137
25 Cross Correlation	139
26 Datapoint Tracer	141
27 Heatmap	143
28 KShape	149
29 Peak Editor	155
30 Proportions	157
31 Scatter	161
32 Simple Plot	163
33 SpaceMap	165
34 Welcome Window	169
35 Project Structure	171
36 Consoles	173
37 Saving plots	175
38 Plot Navbar	177
39 System Configuration	179
40 Nodes	181
41 Plots	189
42 Viewer Modules	193
43 Common	195
44 Viewer Core	199
45 Viewer Modules	203
46 ROI Manager	207
47 Stimulus Mapping	229
48 Data types used for analysis	231
49 Analysis	239
50 Nodes	247
51 Plotting utils	255
52 Plot Bases	257

53 Datapoint Tracer	259
54 Heatmap	261
55 KShape	265
56 Proportions	269
57 Scatter Plot	271
58 SpaceMap	275
59 Indices and tables	277
Python Module Index	279
Index	281

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

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

Questions/Discussion: Gitter room

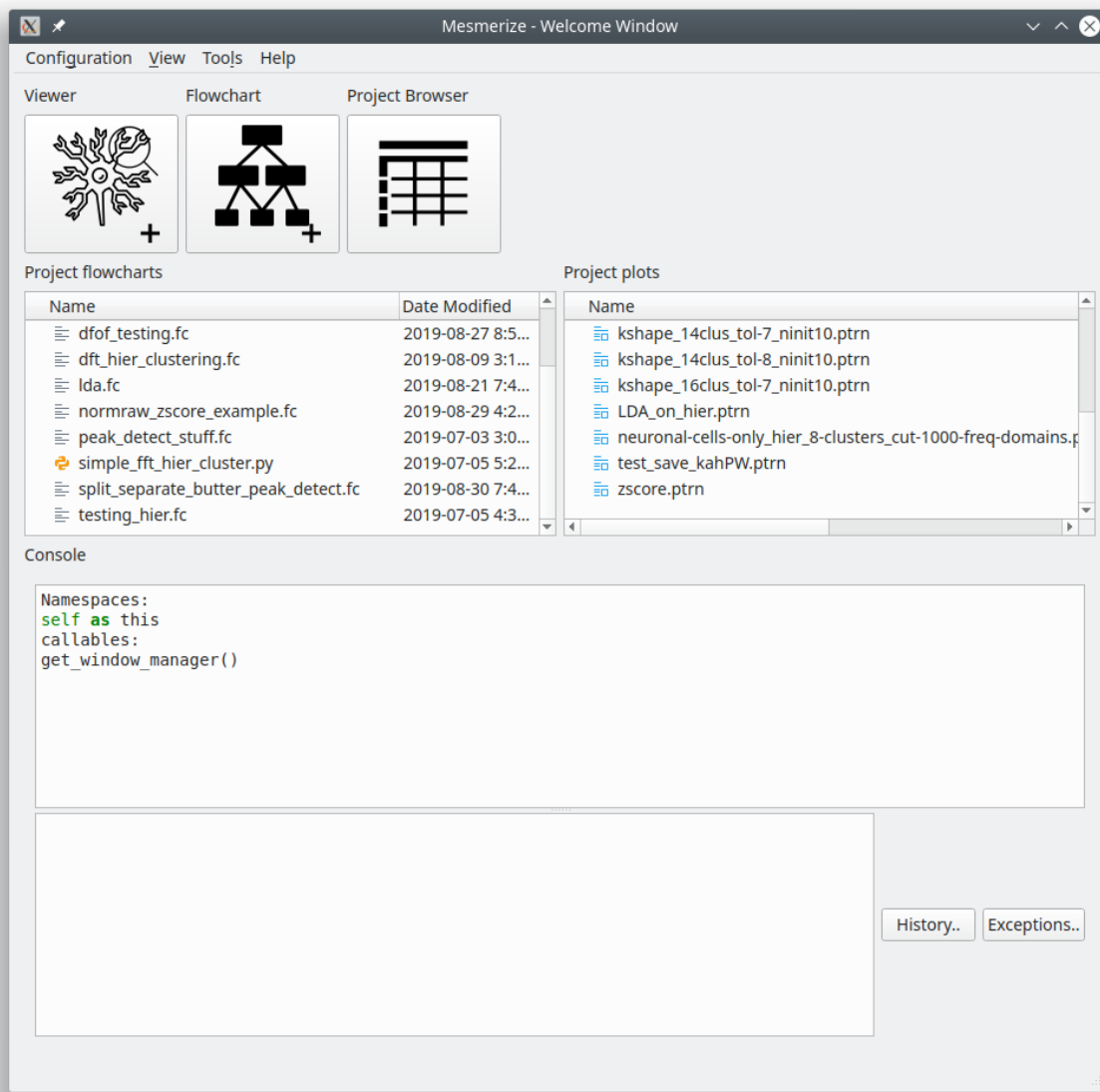
Contact: kushalkolar@alumni.ubc.ca

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.

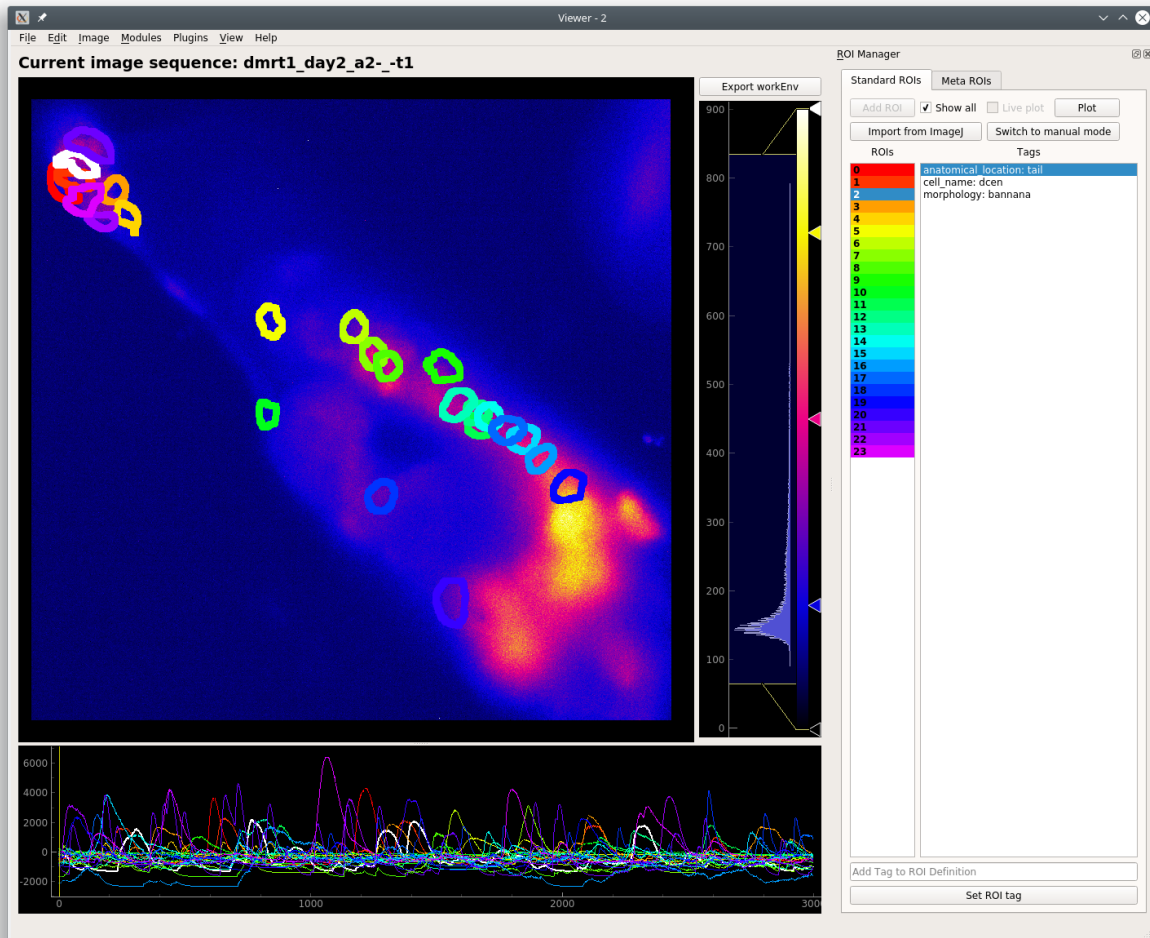
1.1 Welcome Window

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

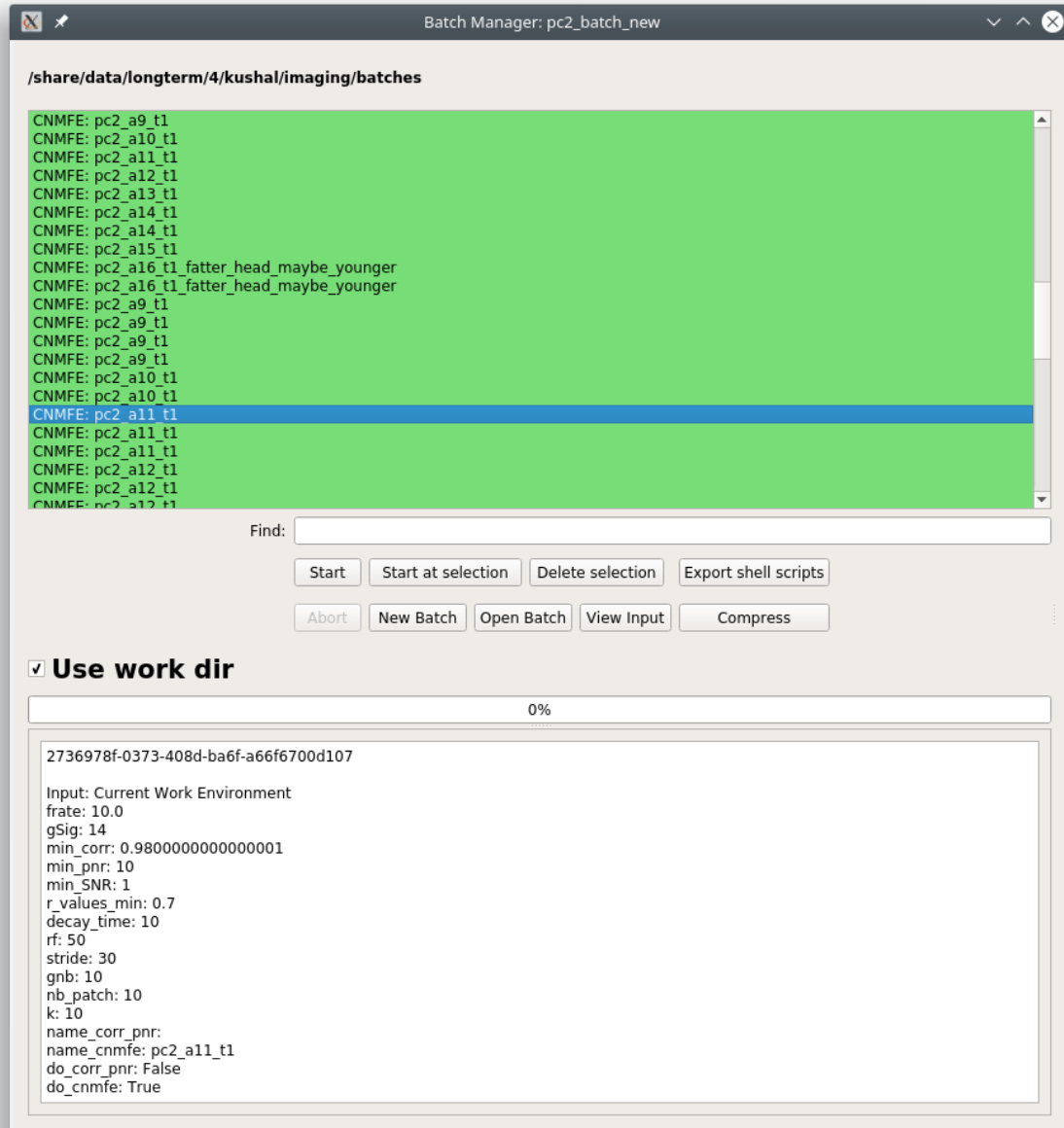
k: 10

Perform CNMF-E: Enter name Add to batch

Export Parameters
Import Parameters

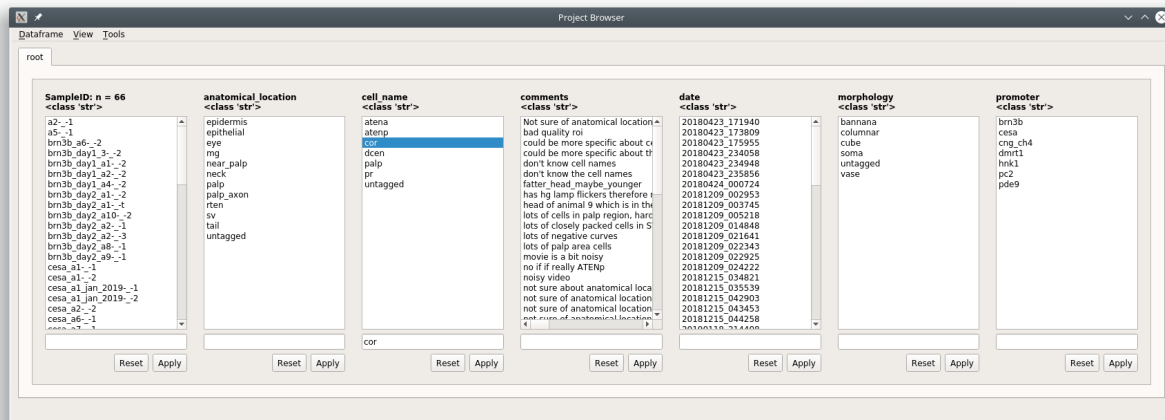
1.4 Batch Manager

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



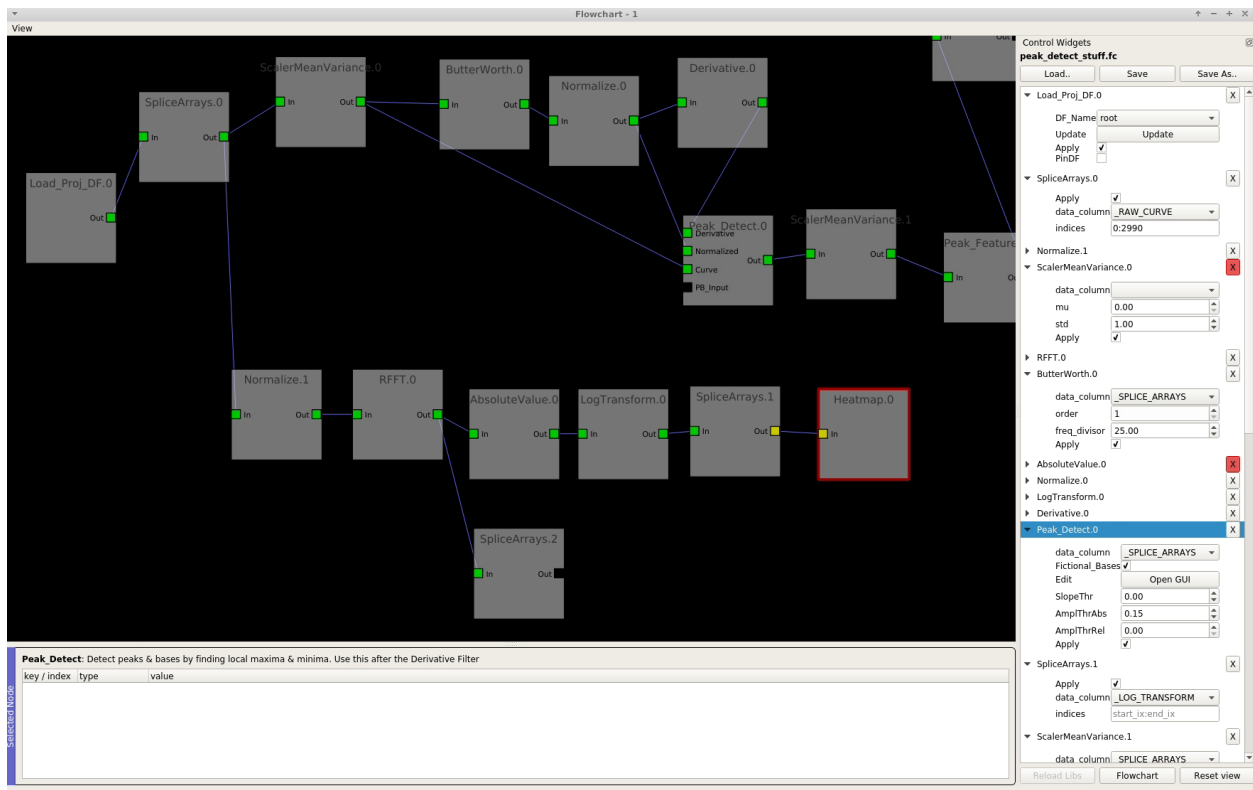
1.5 Project Organization

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

Mesmerize can be installed on Linux, Mac OSX and Windows. On Windows, Mesmerize can be installed in an anaconda environment. For Mac OSX and Linux you may use either virtual environments or conda environments, but we have had much better experience with virtual environments.

2.1 Linux

The snap is currently discontinued in favor of a pypi package.

2.1.1 pip (PyPI)

You will need python==3.6, there is a bug with Qt & python3.7

1. Install python 3.6:

```
# Debian & Ubuntu based
sudo apt-get install python3.6

# Fedora/CentOS
sudo dnf install python36
```

2. Install build tools and other dependencies:

```
# Debian & Ubuntu based distros
sudo apt-get install build-essential python3-devel qt5-default tcl graphviz git

# Fedora/CentOS
sudo dnf install @development-tools
sudo dnf install python3-devel tcl graphviz
```

For other distributions install the equivalent meta package to get build tools.

If you're on Fedora/CentOS you'll also need `redhat-rpm-config`, install using:

```
sudo dnf install redhat-rpm-config
```

1. Create a new virtual environment:

```
python3.6 -m venv <new_venv_path>
```

2. Activate this environment:

```
source <new_venv_path/bin/activate>
```

3. Make sure you have a recent version of pip and setuptools:

```
pip install --upgrade pip setuptools
```

4. Install tslearn & bottleneck (optional):

```
pip install tslearn~=0.2.2 bottleneck==1.2.1
```

5. Install mesmerize:

```
pip install mesmerize
```

6. Now you should be able to launch mesmerize from the terminal:

```
mesmerize
```

You will always need to activate the environment for Mesmerize before launching it.

1. If you want Caiman features you'll need to install caiman into this environment:

```
git clone https://github.com/flatironinstitute/CaImAn
cd CaImAn/
source activate caiman
pip install .
```

More information on caiman installation: <https://caiman.readthedocs.io/en/master/Installation.html#installation-on-macos-and-linux>

2.2 Mac OSX

This requires Anaconda and will install Mesmerize in an Anaconda environment. If you want to install into a python virtual environment use the instructions for the Linux installation from step #3 onward. 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 caiman for Caiman features:

```
conda install -c conda-forge caiman
```

4. Install Mesmerize. On Mac installing tslearn before mesmerize creates problems on anaconda for some reason:

```
pip install mesmerize
```

5. Install cython, and downgrade pandas:

```
conda install Cython pandas~=0.25.3
```

6. Install tslearn (optional):

```
conda install -c conda-forge tslearn==0.2.1
```

7. Install bottleneck (optional):

```
pip install bottleneck==1.2.1
```

8. To launch Mesmerize call it from the terminal:

```
mesmerize
```

You will always need to activate the environment for Mesmerize before launching it.

You might get a matplotlib error similar to below:

```
Bad val 'qt5' on line #1
"backend: qt5

in file "/Users/kushal/.matplotlib/matplotlibrc"
Key backend: Unrecognized backend string 'qt5': valid strings are ['GTK3Agg',
↳ 'GTK3Cairo', 'MacOSX', 'nbAgg', 'Qt4Agg', 'Qt4Cairo', 'Qt5Agg', 'Qt5Cairo', 'TkAgg',
↳ 'TkCairo', 'WebAgg', 'WX', 'WXAgg', 'WXCairo', 'agg', 'cairo', 'pdf', 'pgf', 'ps',
↳ 'svg', 'template']
```

To fix this, 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 Windows

Tested on Windows 10, not sure if it'll work on earlier Windows versions.

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

You will also need git: <https://gitforwindows.org/>

Warning: It is **highly** recommended that you use Mesmerize in a new dedicated environment, even if you already have major dependencies (like caiman) installed in another environment.

All commands are to be run in the powershell

1. You will need anaconda to be accessible through powershell. You may need to run powershell as administrator for this step to to work. Close & open a new non-admin powershell after running this:

```
conda init powershell
```

You will need a relatively recent version of Anaconda in order to run conda commands through the powershell.

1. Create a new anaconda environment:

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

2. Activate the environment:

```
conda activate mesmerize
```

3. Install caiman:

```
conda install -c conda-forge caiman
```

4. Downgrade pandas, install Cython:

```
conda install Cython pandas~=0.25.3
```

5. Install tslearn (optional):

```
conda install -c conda-forge tslearn==0.2.1
```

6. Install bottleneck (optional):

```
pip install bottleneck==1.2.1
```

7. Install graphviz:

```
conda install graphviz
```

8. Install pywin32:

```
pip install pywin32
```

9. Install Mesmerize:

```
pip install mesmerize
```

10. Allow powershell to execute scripts. Run powershell as administrator to execute these commands. This is required for the batch manager and k-Shape GUI which launch external processes. This may affect the security of your system by allowing scripts to be executable. I'm not an expert on Windows so if someone knows a better way to do this let me know! As far as I know, I'm not sure why you would try to execute untrusted scripts so this shouldn't be a concern?:

```
Set-ExecutionPolicy RemoteSigned  
Set-ExecutionPolicy Bypass -scope Process -Force
```

11. Launch Mesmerize:

```
mesmerize
```

Note: In order to use some features, such as the batch manager, you will need to check your *System Configuration settings in Mesmerize* to make sure that it activates the conda environment that mesmerize is installed in. By default the

pre-run commands contain `# conda activate mesmerize` but you will need to uncomment the line (remove the `#`) or change it if you are using an environment with a different name.

2.4 From GitHub (Development)

First, make sure you have compilers & python3.6 (see the details above for various Linux distros or Mac OSX)

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. Upgrade pip & setuptools & install some build dependencies:

```
pip install --upgrade pip setuptools
pip install Cython numpy tslearn==0.2.2
```

4. Fork the main repo on github and clone it:

```
git clone https://github.com/<your_github_username>/MESmerize.git
cd MESmerize
```

5. Switch to new branch:

```
git checkout -b my-new-feature
```

6. Install in editable mode:

```
pip install -e .
```

7. Make your changes to the code & push to your fork:

```
git push origin my-new-feature
```

8. Create a pull request if you want to incorporate it into the main Mesmerize repo.

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 may need to separate the CNMF(E) and manual data in downstream analysis if using flowchart nodes that work with data from specific sources.

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.

Note: It is generally advisable to keep the names of your categorical data columns short with lowercase letters. When sharing your project you can provide a mapping for all your keys. This helps maintain consistency throughout your project and makes the data more readable.

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 the 'Columns' configuration window with the following sections:

- Include in Project Browser View:** SampleID, date, comments, anatomical_location, odor, gene.
- Exclude in Project Browser View:** CurvePath, ImgInfoPath, ImgPath, MaxProjPath, ROI_State, uuid_curve, misc.
- ROI Type Columns:** anatomical_location.
- Stimulus Type Columns:** odor.
- Custom Columns:** gene.
- Add new custom column:** A text input field containing 'developmental_stage'.
- Choose data type:**
 - ☐ standard text (str)
 - ☒ whole numbers (np.int64)
 - ☐ decimal numbers (np.float64)
 - ☐ boolean (True/False values)
- Dataframe replacement value:** An empty text input field.
- Buttons:** 'Add' (next to the new column input), '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:

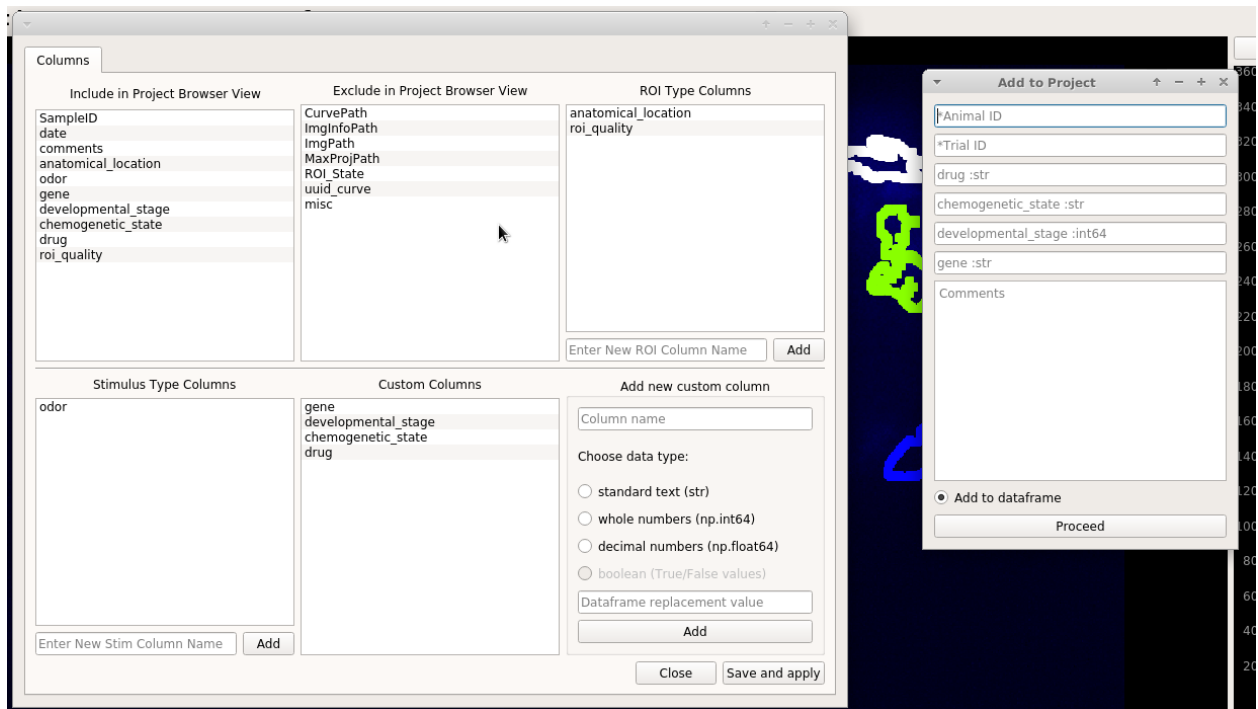
- Include in Project Browser View:** SampleID, date, comments, anatomical_location, odor.
- Exclude in Project Browser View:** CurvePath, ImgInfoPath, ImgPath, MaxProjPath, ROI_State, uuid_curve, misc.
- ROI Type Columns:** anatomical_location. Below this is a text input 'Enter New ROI Column Name' and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

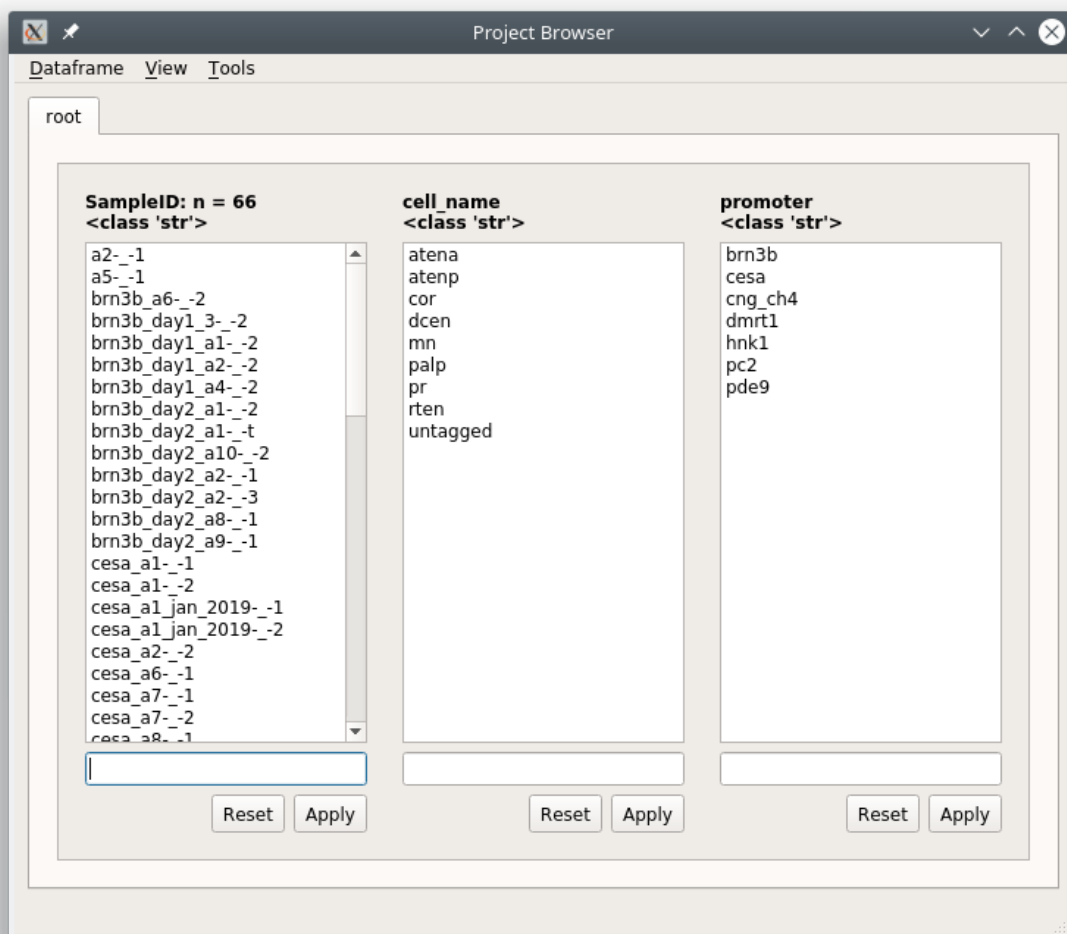
See also:

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.



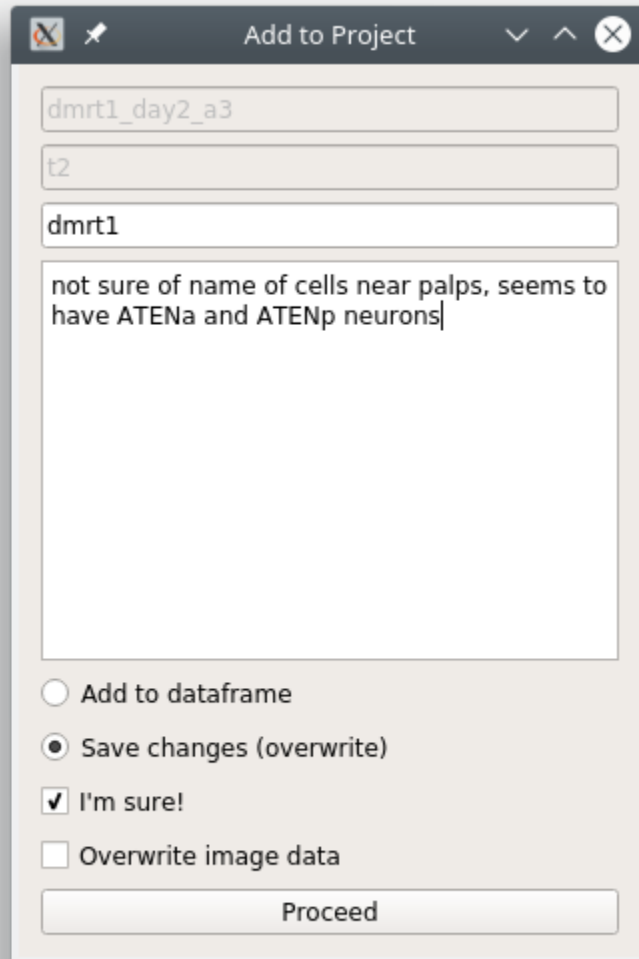
The columns that are visible in the Project Browser Window correspond to the *Project Configuration*. For each column you will see a list which is a set of unique elements from that column in the project DataFrame.

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

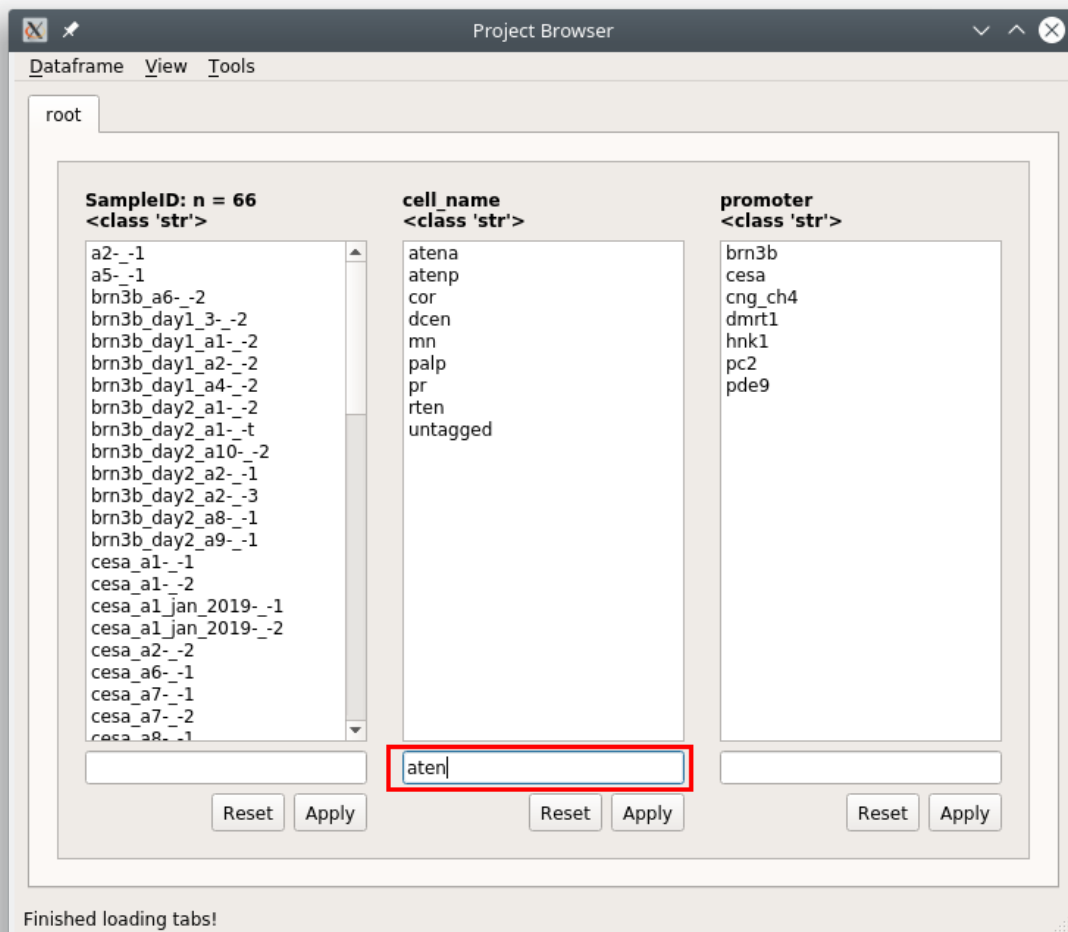
You can sort your Project DataFrame into different groups (such as experimental groups) using text and numerical filters. Type a filter into the text entries that are below the list of elements for a column. You can also click on one or many elements in a column to set those elements as the filters.

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”

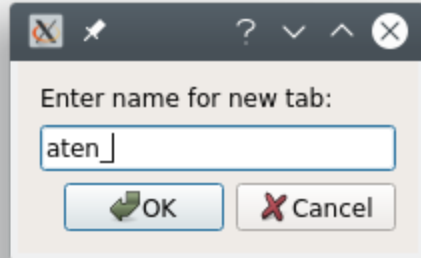
6.2.1 Text filters

6.2.2 Partial match

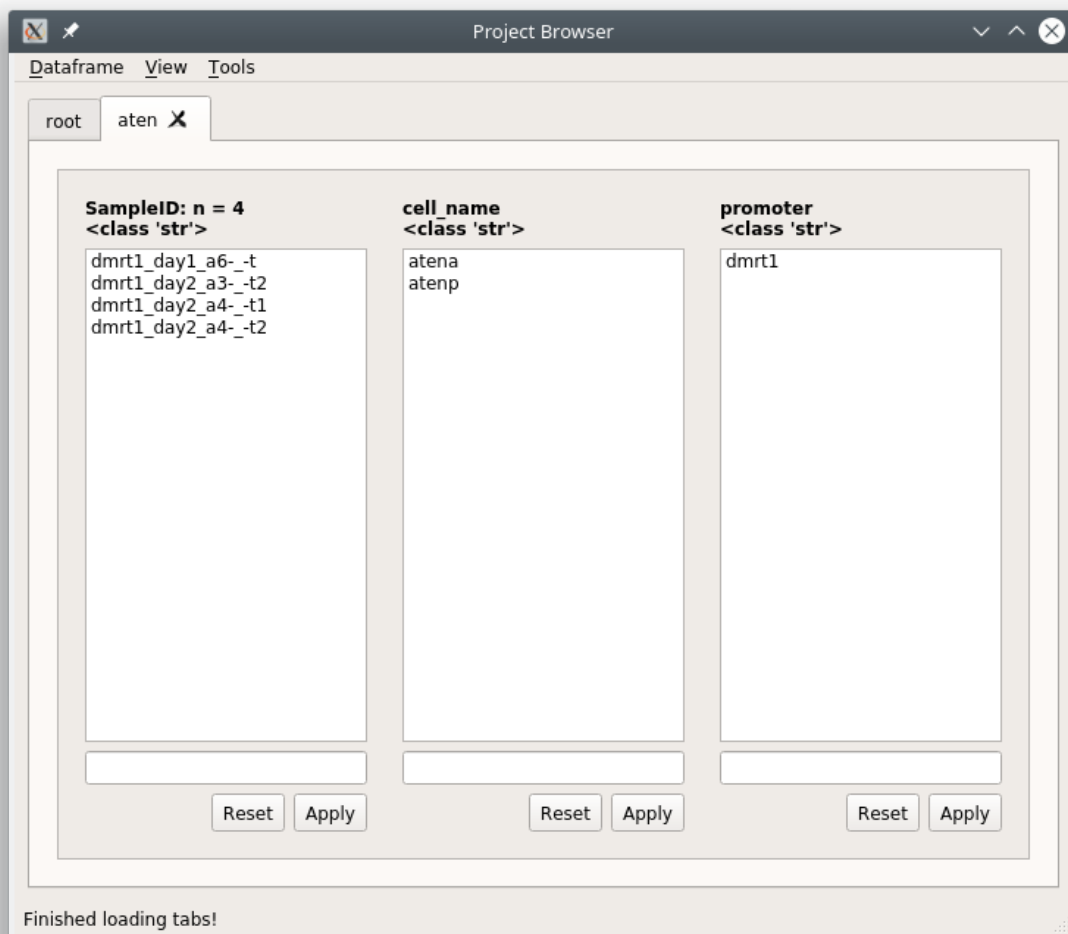
To filter out a group based on partial text matches just enter the text into the filter text entry below the column(s) of interest and click “Apply”



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.



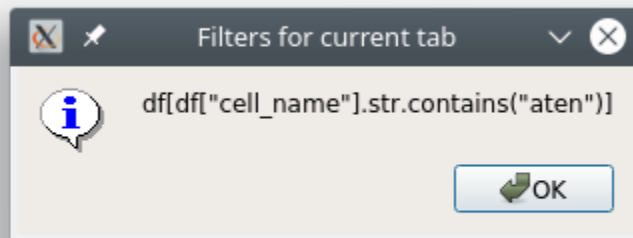
The result is a DataFrame containing all rows where the cell_name contains 'aten'



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

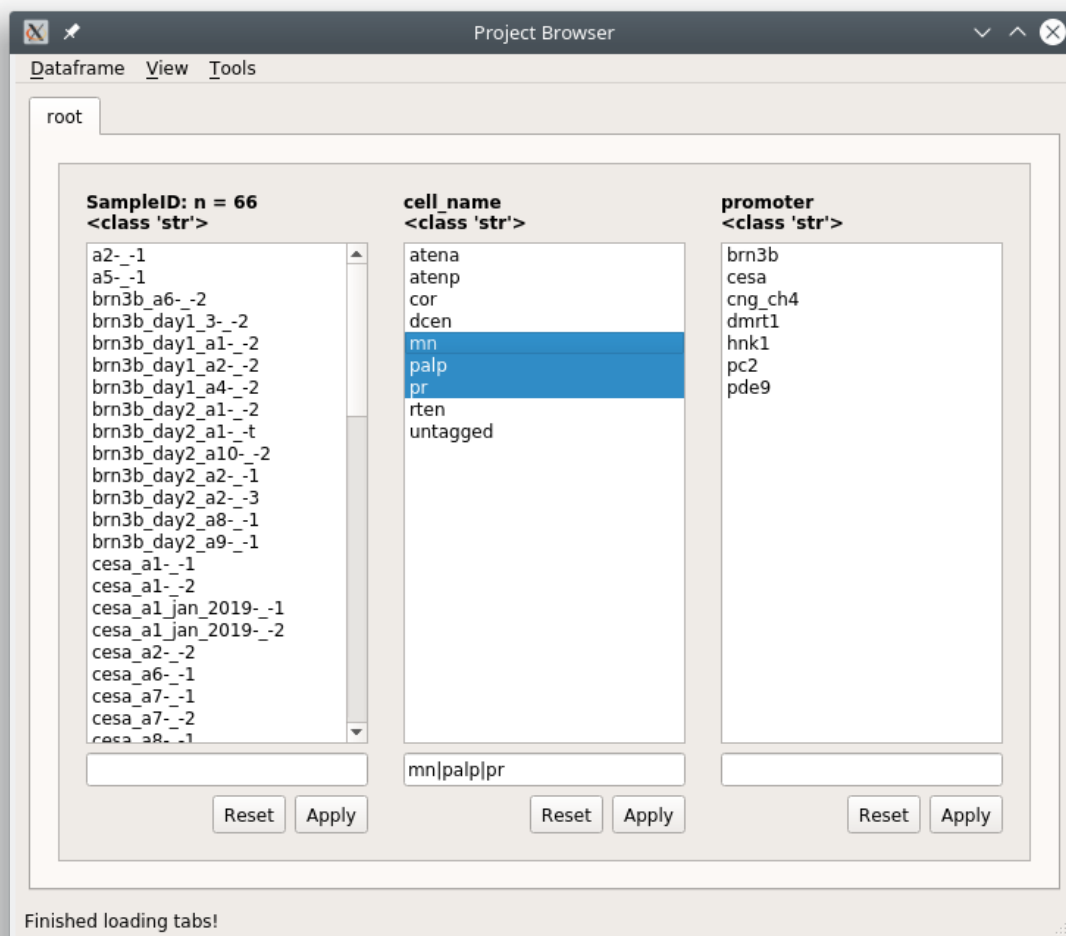
Index	CurvePath	ImgInfoPath	ImgPath	ImgUUID	ROI_State	SampleID	anatomical_locati	cell_name	comments	date	misc	morphology	promoter	stimulus_name	uuid_curve
2	curves/dnr...	images/dnr...	images/dnr...	1b2b1436-0...	{'roi_xs': '...	dnrt1_day2...	sv	atena	not sure o...	20190427_0...	()	untagged	dnrt1	['untagged...	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

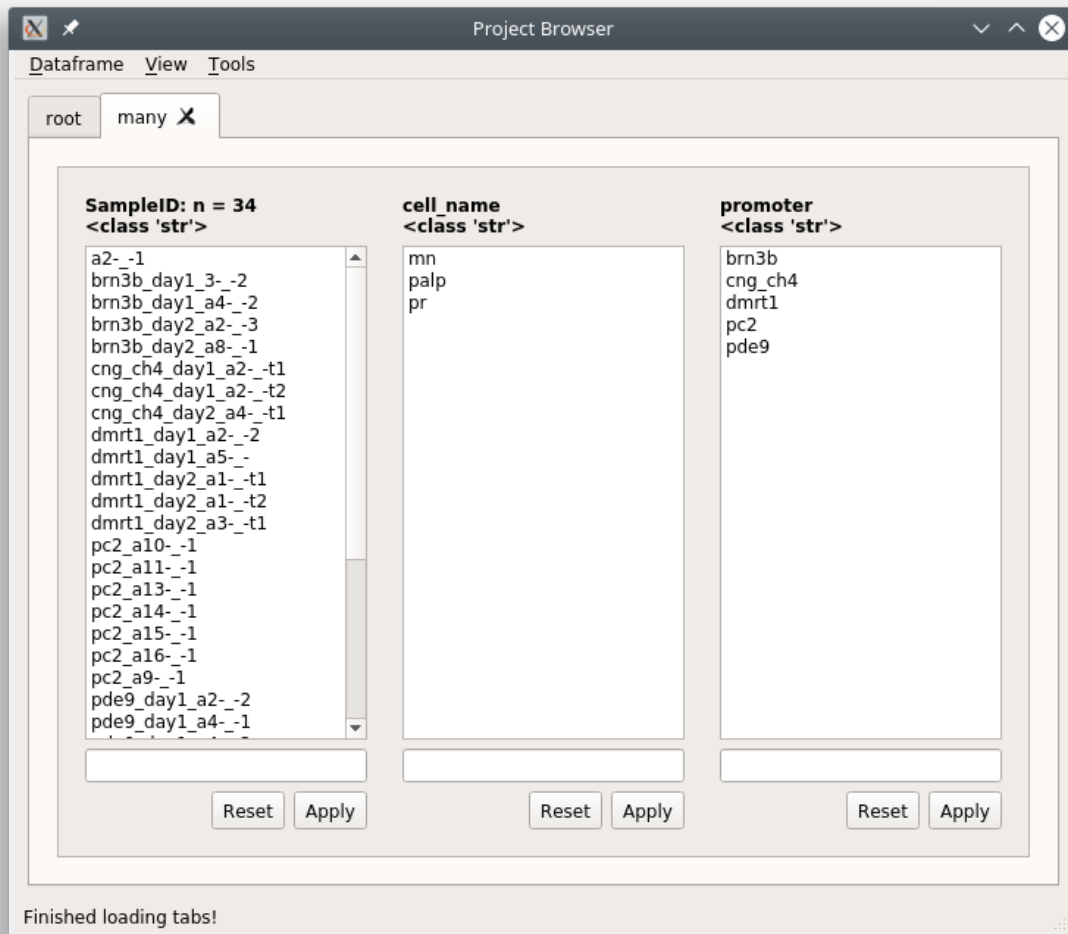


6.2.3 Multiple filters

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.

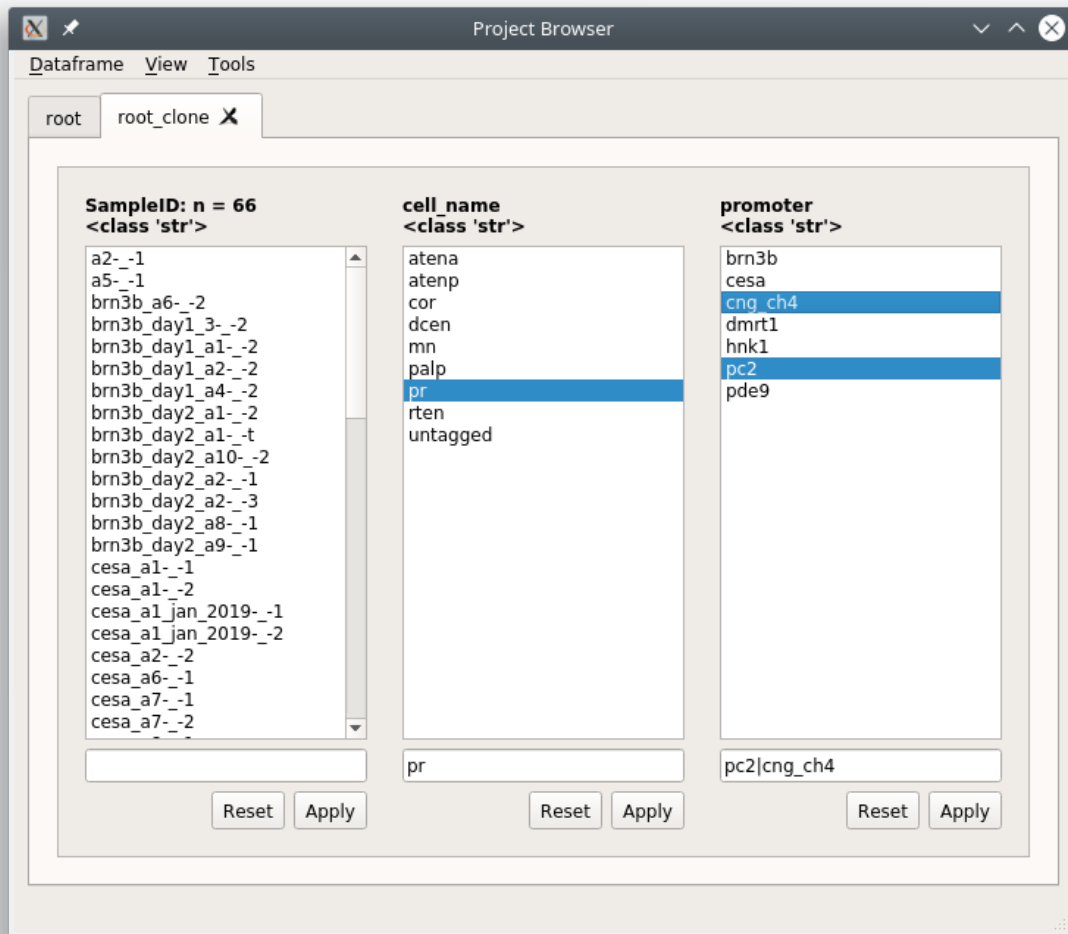


Note: This can be combined with *Modifiers*

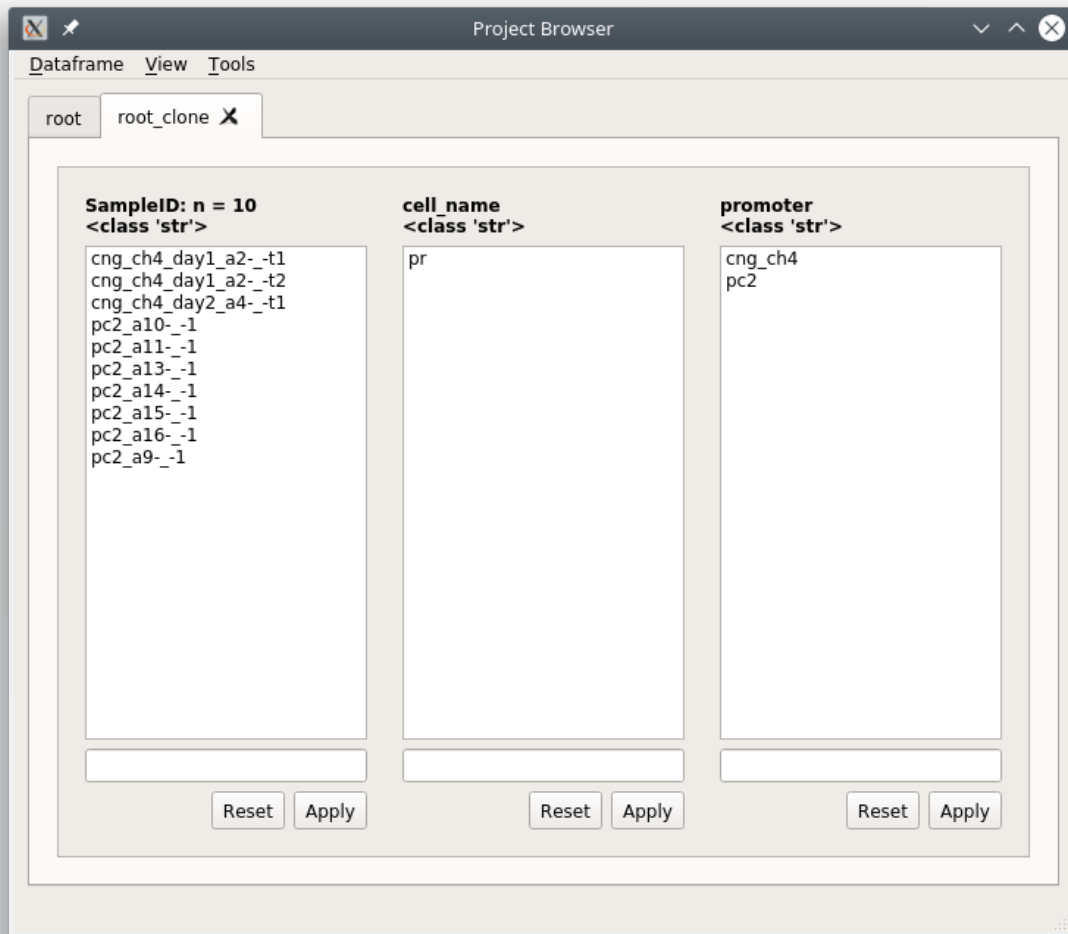
6.2.4 Filter multiple columns

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”.

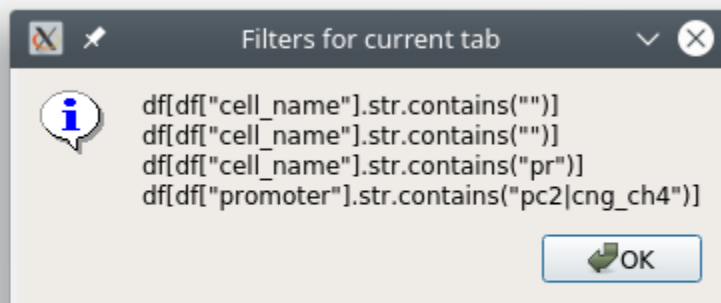
Filter out all rows where the cell_name column contains ‘pr’ and promoter column contains ‘pc2’ or ‘cng_ch4’.



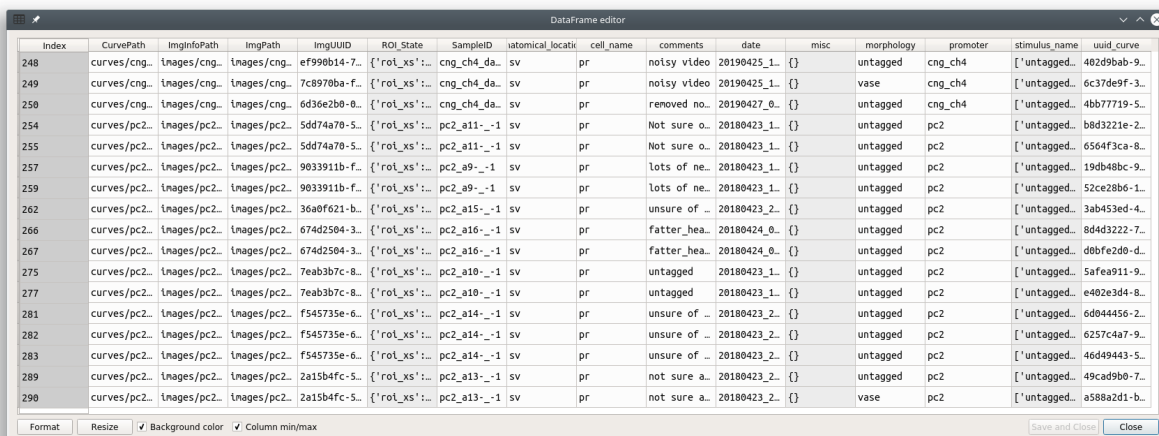
Right click on the “Apply” button and choose “Apply all” or “Apply all in new tab”



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)
```

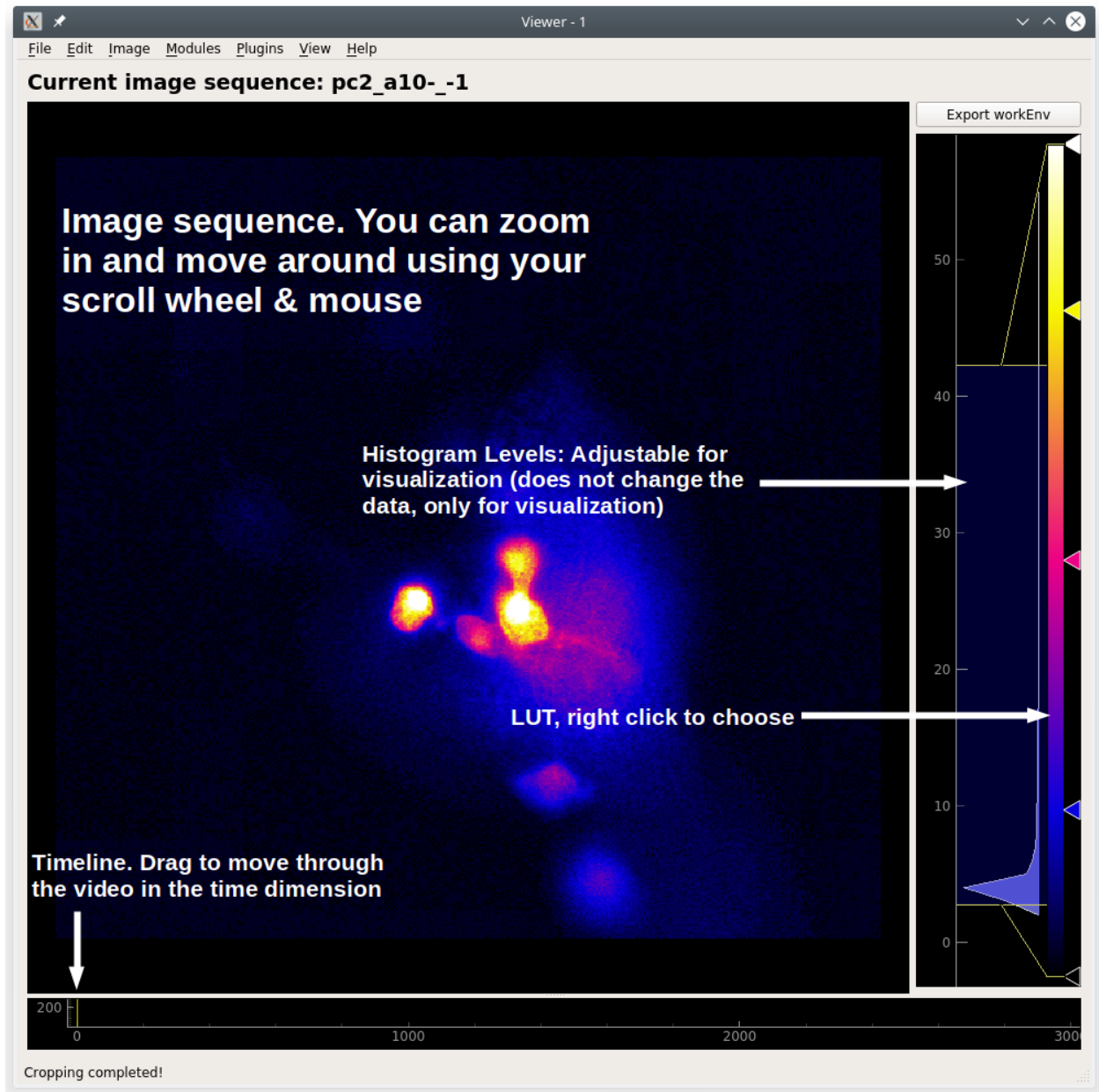
VIEWER OVERVIEW

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.

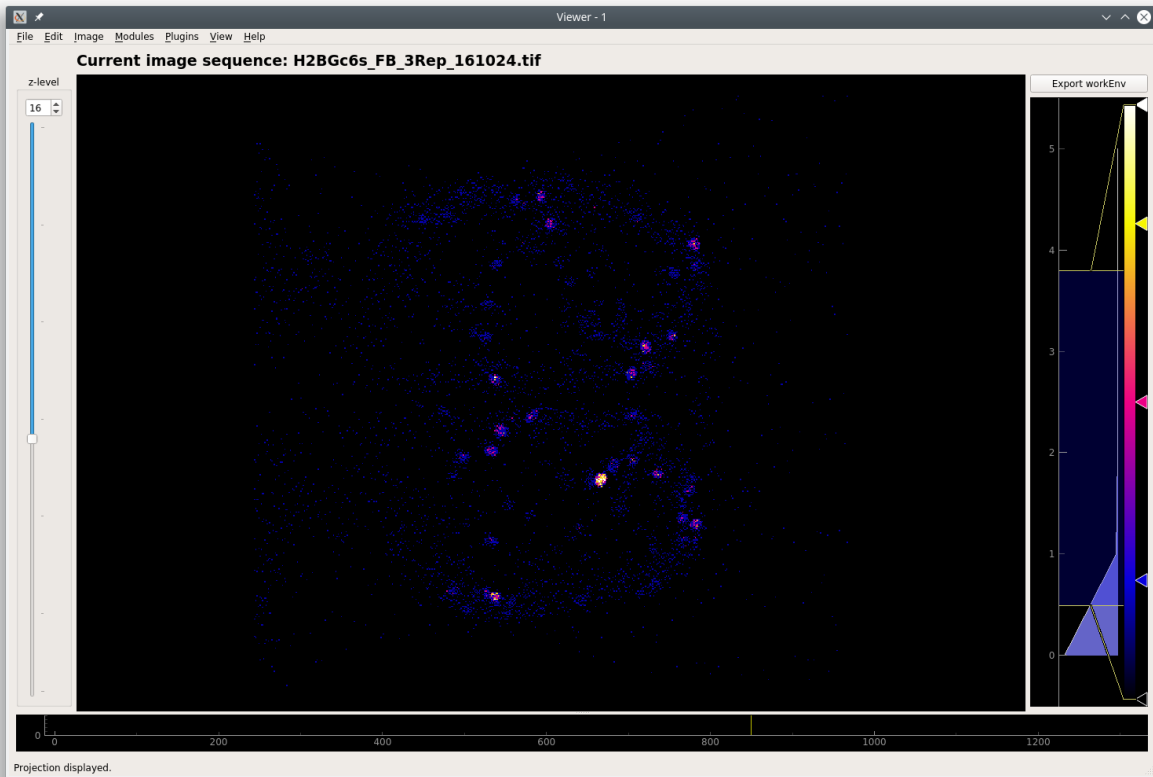
7.1 Layout



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.

7.1.1 3D data

When viewing 3D data a slider on the left allows you to move through the z axis.



The image stack shown above is from Martin Haesemeyer's dataset from the following paper:

Haesemeyer M, Robson DN, Li JM, Schier AF, Engert F. A Brain-wide Circuit Model of Heat-Evoked Swimming Behavior in Larval Zebrafish. *Neuron*. 2018;98(4):817-831.e6. doi:10.1016/j.neuron.2018.04.013

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	<i>ImgData</i> object containing the Image Sequence and meta data from the imaging source
roi_manager	The back-end <i>ROI Manager</i> 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>CNMF</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 VBox

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. If the data are 3D, the projection is of the current plane.

7.3.4 Modules

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

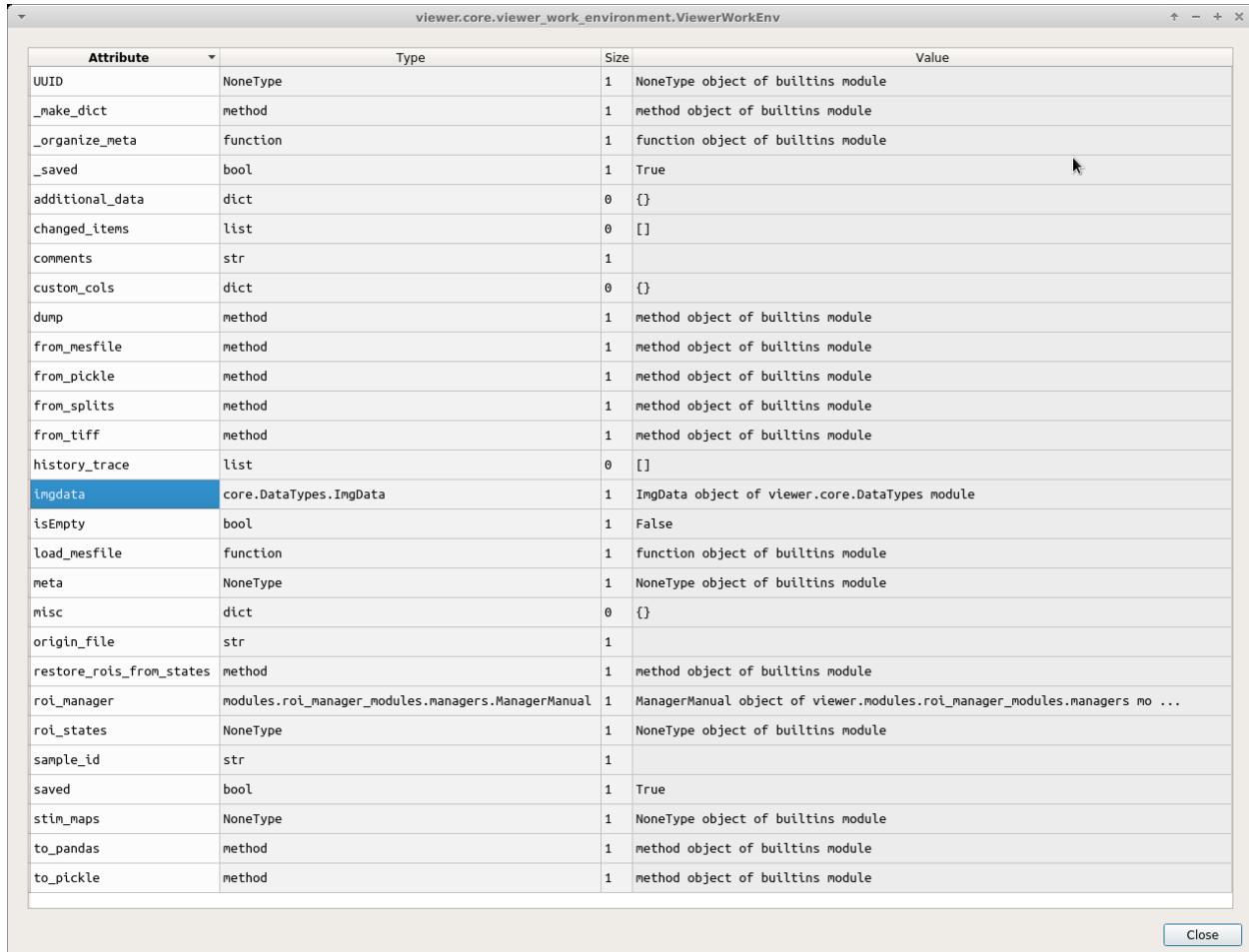
7.3.5 Plugins

Custom viewer modules.

7.3.6 View

Work Environment Editor

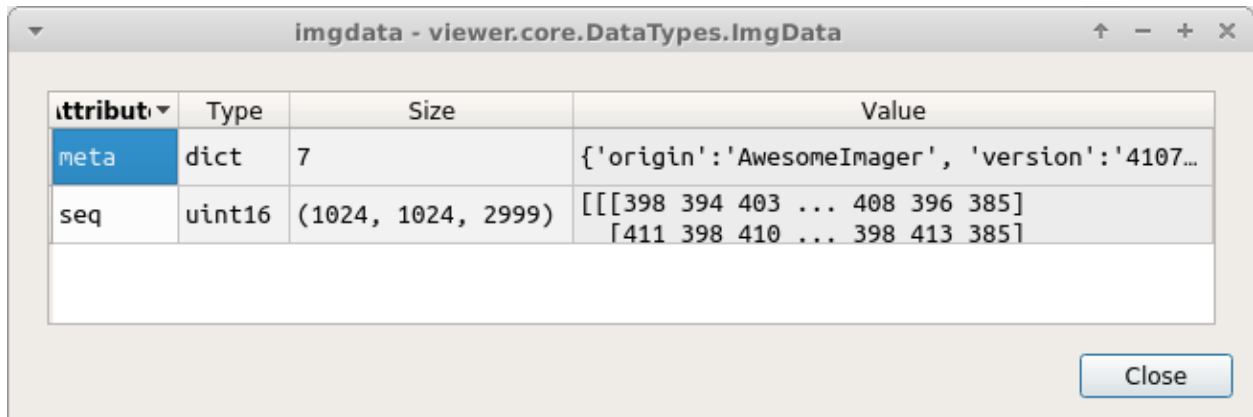
Explore the data in your work environment using a GUI.



Attribute	Type	Size	Value
UUID	NoneType	1	NoneType object of builtins module
_make_dict	method	1	method object of builtins module
_organize_meta	function	1	function object of builtins module
_saved	bool	1	True
additional_data	dict	0	{}
changed_items	list	0	[]
comments	str	1	
custom_cols	dict	0	{}
dump	method	1	method object of builtins module
from_mesfile	method	1	method object of builtins module
from_pickle	method	1	method object of builtins module
from_splits	method	1	method object of builtins module
from_tiff	method	1	method object of builtins module
history_trace	list	0	[]
imgdata	core.DataTypes.ImgData	1	ImgData object of viewer.core.DataTypes module
isEmpty	bool	1	False
load_mesfile	function	1	function object of builtins module
meta	NoneType	1	NoneType object of builtins module
misc	dict	0	{}
origin_file	str	1	
restore_rois_from_states	method	1	method object of builtins module
roi_manager	modules.roi_manager_modules.managers.ManagerManual	1	ManagerManual object of viewer.modules.roi_manager_modules.managers mo ...
roi_states	NoneType	1	NoneType object of builtins module
sample_id	str	1	
saved	bool	1	True
stim_maps	NoneType	1	NoneType object of builtins module
to_pandas	method	1	method object of builtins module
to_pickle	method	1	method object of builtins module

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

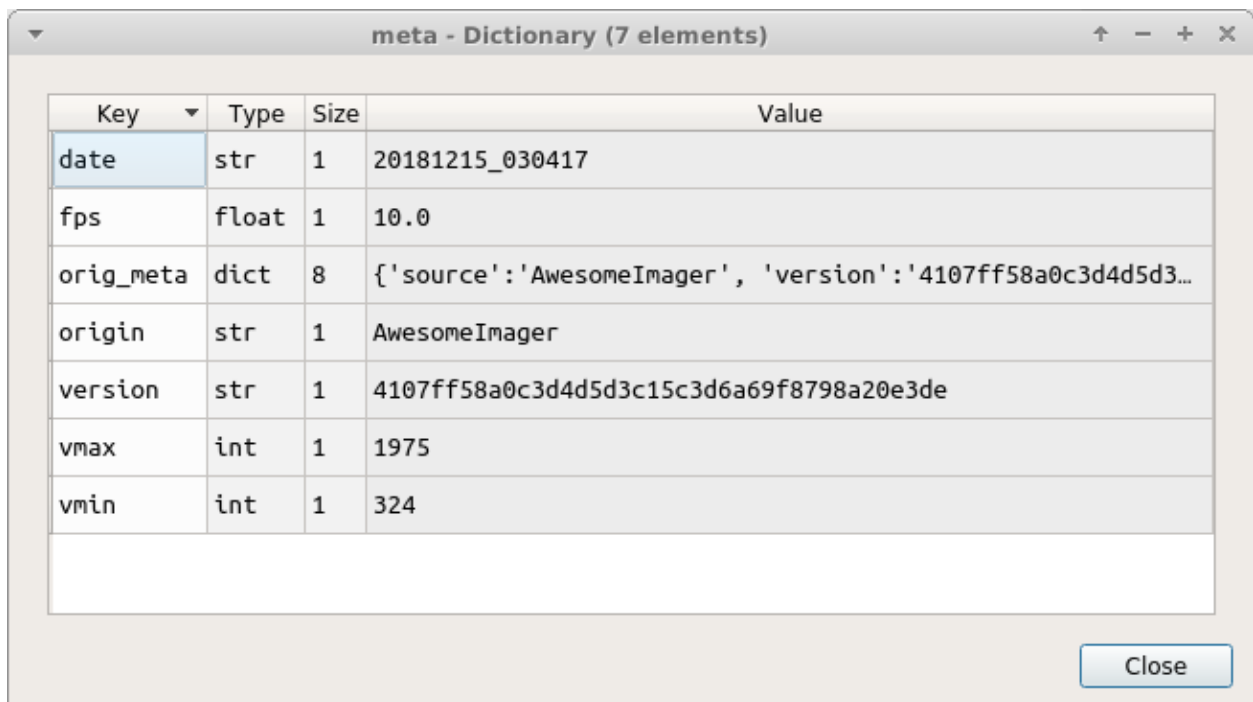
For example if you want to see your meta data, double click on “imgdata” and then you can see that “imgdata” has two things, the image sequence (i.e. your video) and the meta data.



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.

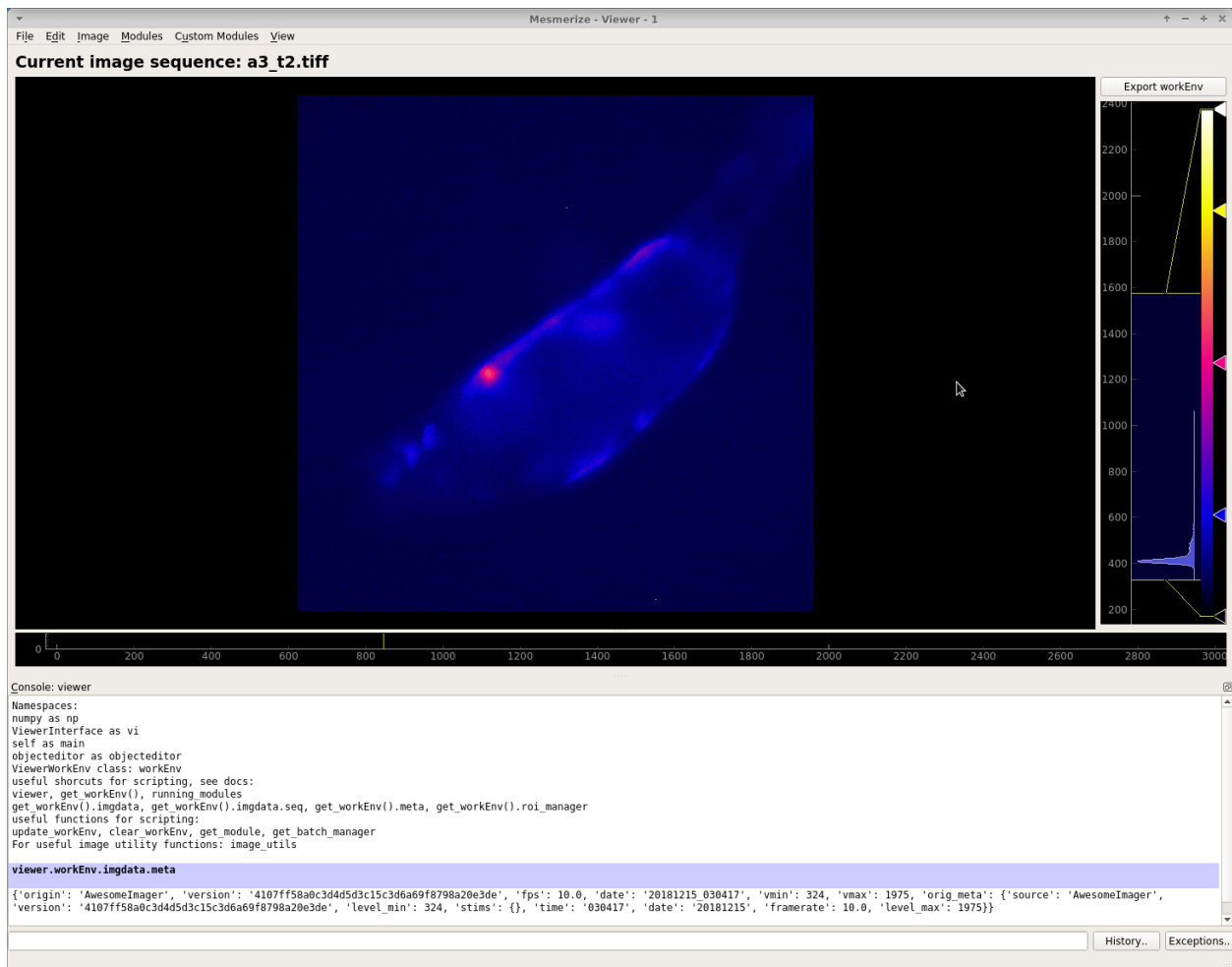


Key	Type	Size	Value
date	str	1	20181215_030417
fps	float	1	10.0
orig_meta	dict	8	{'source': 'AwesomeImager', 'version': '4107ff58a0c3d4d5d3...
origin	str	1	AwesomeImager
version	str	1	4107ff58a0c3d4d5d3c15c3d6a69f8798a20e3de
vmax	int	1	1975
vmin	int	1	324

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 <i>ViewerUtils</i> . Use this to interact with the viewer.
all_modules	List all available modules (includes default and any available plugins/custom modules)
Viewer-WorkEnv	Use for creating new instances of <i>ViewerWorkEnv</i>
ImgData	Use for creating new instances of <i>ImgData</i>
get_workEnv()	Get the current viewer <i>work environment</i> (instance of <i>ViewerWorkEnv</i>)
get_image()	Get the current image sequence (returns current <i>ViewerWorkEnv.imgdata.seq</i>). If the data are 3D it returns the current plane only.
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
↳ (continues on next page)
↳ '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

Use the :ref:`Viewer Core API <API_ViewerCore> to open any arbitrary image

This example loads an image stored using `numpy.save()`, but this is applicable to images stored in any format that can eventually be represented as a numpy array in python. For example, you could also load image files stored in HDF5 format and load the numpy array that represents your image sequence.

```

1  import numpy as np
2
3  # clear the viewer work environment
4  clear_workEnv()
5
6  a = np.load('/path_to_image.npy')
7
8  # check what the axes order is
9  a.shape
10
11 # (1000, 512, 512) # for example
12 # looks like this is in [t, x, y]
13 # this can be transposed so we get [x, y, t]
14 # ImgData takes either [x, y, t] or [x, y, t, z] axes order
15
16 # Define a meta data dict
17 meta = \
18     {
19         "origin":      "Tutorial example",
20         "fps":          10.0,
21         "data":         "20200629_171823",
22         "scanner_pos": [0, 1, 2, 3, 4, 5, 6]
23     }
24
25 # Create ImgData instance
26 imgdata = ImgData(a.T, meta) # use a.T to get [x, y, t]
27
28 # Create a work environment instance
29 work_env = ViewerWorkEnv(imgdata)
30
31 # Set the current Viewer Work Environment from this new instance
32 vi.viewer.workEnv = work_env
33
34 # Update the viewer with the new work environment
35 # this MUST be run whenever you replace the viewer work environment (the previous_
36 ↪line)
37 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.

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.

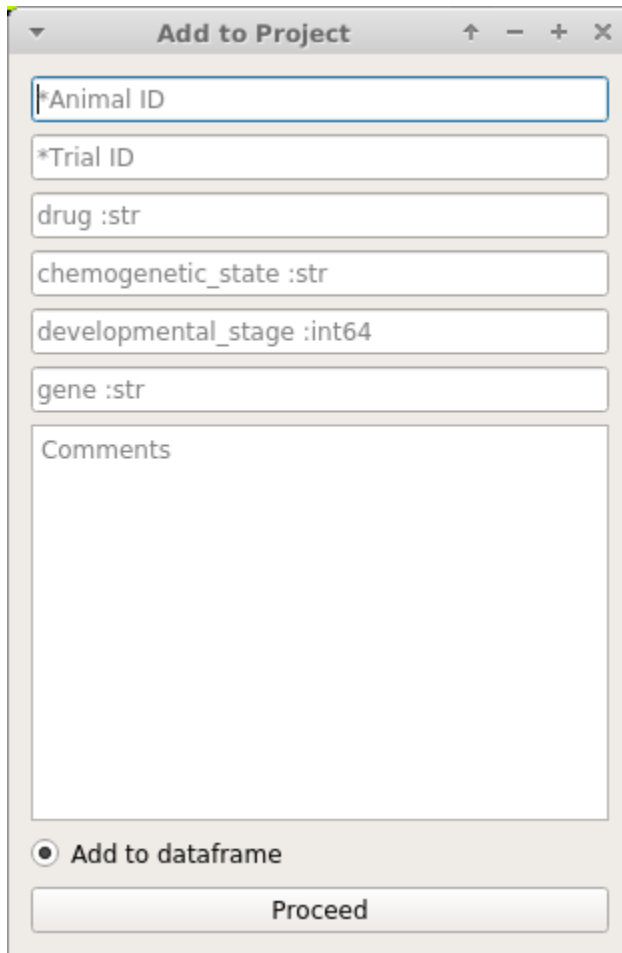
Each sample in your project contains the following:

- 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

8.1 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 stacked vertically: a text field for "*Animal ID", a text field for "*Trial ID", a text field for "drug :str", a text field for "chemogenetic_state :str", a text field for "developmental_stage :int64", and a text field for "gene :str". Below these fields is a large, empty text area labeled "Comments". At the bottom of the window, there is a radio button next to the text "Add to dataframe", which is currently selected. Below the radio button is 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 [Tiff module API](#).

To open tiff files first click the “Select file” button and choose your file. You can also drag and drop a tiff file (drag and drop doesn’t work properly on Windows).

The screenshot shows a software window titled "Tiff file I/O". It is divided into two main panels. The left panel, titled "Tiff File", contains a "Select tiff file" button, a section for "Image Load Method:" with three radio button options: "asarray", "asarray - multi series", and "imread", and a section for "Image Axes order" with two radio button options: "2D: [t, x, y]" (which is selected) and "3D: [t, z, x, y]", plus a "Custom, ex: xyz" text input field. The right panel, titled "Meta data file", contains a checked checkbox for "Load meta data", a "Select meta data file" button, and a "Meta data format:" section with a text area containing "Awesomemager" and "json_minimal". At the bottom of the window is a large "Load into Work Environment" button.

Next, you must select an appropriate Image Load Method (see next section). You can also import meta data associated with your recording.

Certain meta data, such as the sampling rate of the data, are necessary for some downstream analysis procedures. There are a few ways to import your meta data into the Viewer Work Environment:

- Simple JSON files, see *json_minimal* under the table in the [Meta data](#) section

- Define your own *Custom functions* to open meta in other file formats
- Manually create a meta data dictionary using the *Console*

9.1 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, [create an issue on GitHub](#).

- **asarray**: Should work for most tiff files, fast method
- **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.
- **imread**: Usually slower, should work for most tiff files.

9.2 Axes order

Choose the default axes order or manually enter the axes order if your tiff file uses a different order.

9.3 Meta data

Check the “Load meta data” checkbox if you want to load meta data. Alternatively, you can uncheck this box and create a meta data dictionary manually using the console (see the *Console* section)

You can select a meta data format from the list. This list of formats correspond to the functions available in the module: `mesmerize.viewer.core.organize_meta`. When you select a meta data format, it will automatically try to find a file with the extension specified by the selected format if it has the same name as the selected tiff file.

If you have questions on meta data formats feel free to drop a message in the [Gitter room](#)

Default list of formats that are recognized:

Name	extension	Description
json_minimal	.json	Recognizes a json file that contains at least the minimal set of necessary keys: <code>origin</code> , <code>fps</code> and <code>date</code> . All other keys in the JSON file are placed in a sub-dictionary with the key <code>orig_meta</code> See <i>Minimal dict</i> below for more info.
AwesomeImager	.json	Used for 1p imaging in the Chatzigeorgiou group at the Sars Center

9.3.1 Custom functions

You may define your own function to organize your meta data. It **MUST** return a dict which has at least the following keys: `origin`, `fps` and `date`.

- `origin` is a `str` describing the software or microscope the recording comes from. This is for your own record.
- `fps` is the sampling rate of the recording as a `float` or `int`
- `date` is the date & time represented by a `str` in the following format: `"YYYYMMDD_HHMMSS"`

In addition to these 3 keys, you may include any additional keys as you wish.

If you think your meta data organizing function will be useful for others I'll be happy to review a pull request and it can be included by default in Mesmerize. We're happy help you create a meta data function, just contact us on [Gitter](#) or [create an issue on GitHub](#).

Minimal dict

Example of a minimal meta data dict.

```
{
    'origin': "microscope or software origin", # must be a str
    'fps':    10.0,                          # must be a int or float
    'date':   "20201123_172345"               # must be a str formatted as
    ↪ "YYYYMMDD_HHMMSS"
}
```

Function outline

Basic outline of a function that you can add to `mesmerize.viewer.core.organize_meta` for organizing your meta data:

1. The function can only take the path to the meta data file as the argument.
2. The expected file extension for the meta data must be specified. The files of a single format are allowed to have multiple different file extension but you must only specify the most common one.
3. The function would generally open the meta data file specified by the `path`, using any python libraries or other code of your choice, and finally return a dictionary that contains the minimal complement of keys, i.e. `origin`, `fps` and `date` with values of the appropriate types (see previous section).

```
def my_meta_organizer(path: str) -> dict:
    """.ext""" # define the file ext in the docstring

    raw_meta = function_to_load_my_file(path)

    # do stuff to organize the raw_meta

    meta = ... # stuff to organize raw meta
    return meta
    # return the organized meta data dict
    # that mesmerize can use
```

9.4 Console/Script usage

You can also load tiff files through the *Viewer Console* or *Script Editor*.

This example can be run line-by-line through the *Viewer Console*, or from the *Script Editor*.

```
1 image_path = # path to tiff file
2 meta_path = # path to json meta data file
3
4 clear_workEnv() # Prevents a confirmation dialog from appearing
5
6 # Get the tiff module
7 tio = get_module('tiff_io', hide=True)
8
9 # Load the tiff file
10 tio.load(image_path, method='imread', axes_order='txy', meta_path=meta_path, meta_
    ↪ format='json_minimal')
```

Alternatively, you may manually create a meta data dictionary after loading a tiff file:

```
1 image_path = # path to tiff file
2
3 clear_workEnv() # Prevents a confirmation dialog from appearing
4
5 # Get the tiff module
6 tio = get_module('tiff_io', hide=True)
7
8 # Load the tiff file
9 tio.load(image_path, method='imread', axes_order='txy')
10
11 meta_dict = \
12     {
13         "origin":    "my_microscope_software",    # must a str
14         "fps":       17.25,                        # must be a int or float
15         "date":      "20201123_172345"             # must be a str formatted as
    ↪ "YYYYMMDD_HHMMSS"/
16     }
17
18 get_workEnv().imgdata.meta = meta_dict
```

See also:

Tiff module API, Viewer Core API, Overview on consoles

BATCH MANAGER

Batch process computationally intensive tasks.

See also:

Batch Manager API

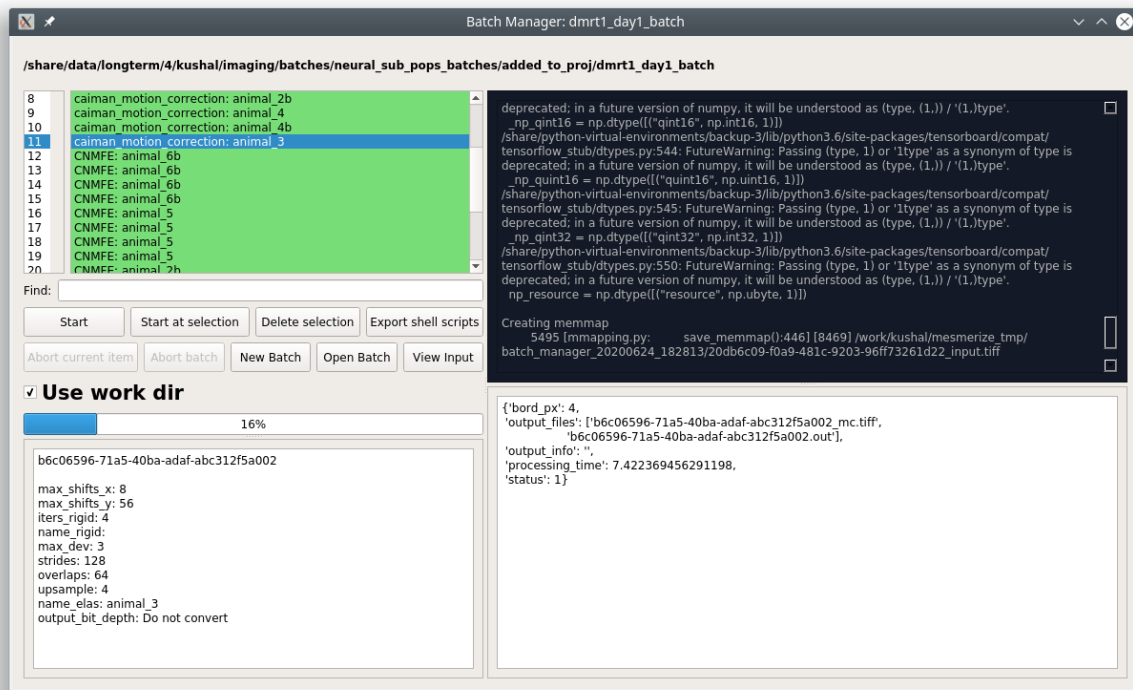
This is currently used for *Caiman Motion Correction*, *CNMF*, *CNMF 3D* and *CNMF-E*.

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.

10.1 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 current item	Abort the current batch item and move on to the next item
Abort batch	Abort the current item and stop processing the batch
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

Use work dir: Check this box to use the work dir that has been set in the *System Configuration*. This feature is only available on Linux & Mac OSX.

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.

10.2 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

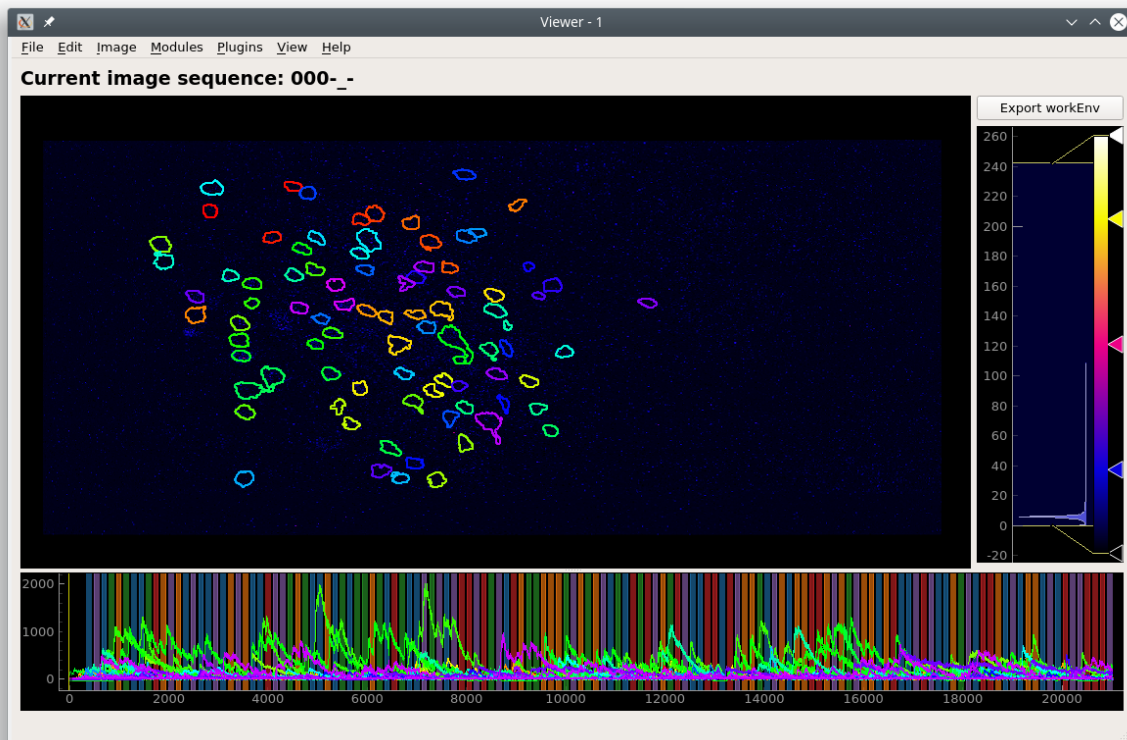
Show on timeline:

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:

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.

11.1 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.

Note: It is generally advisable to keep your stimulus names short with lowercase letters. When sharing your project you can provide a mapping for all your keys. This helps maintain consistency throughout your project and makes the data more readable.

11.2 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`

```

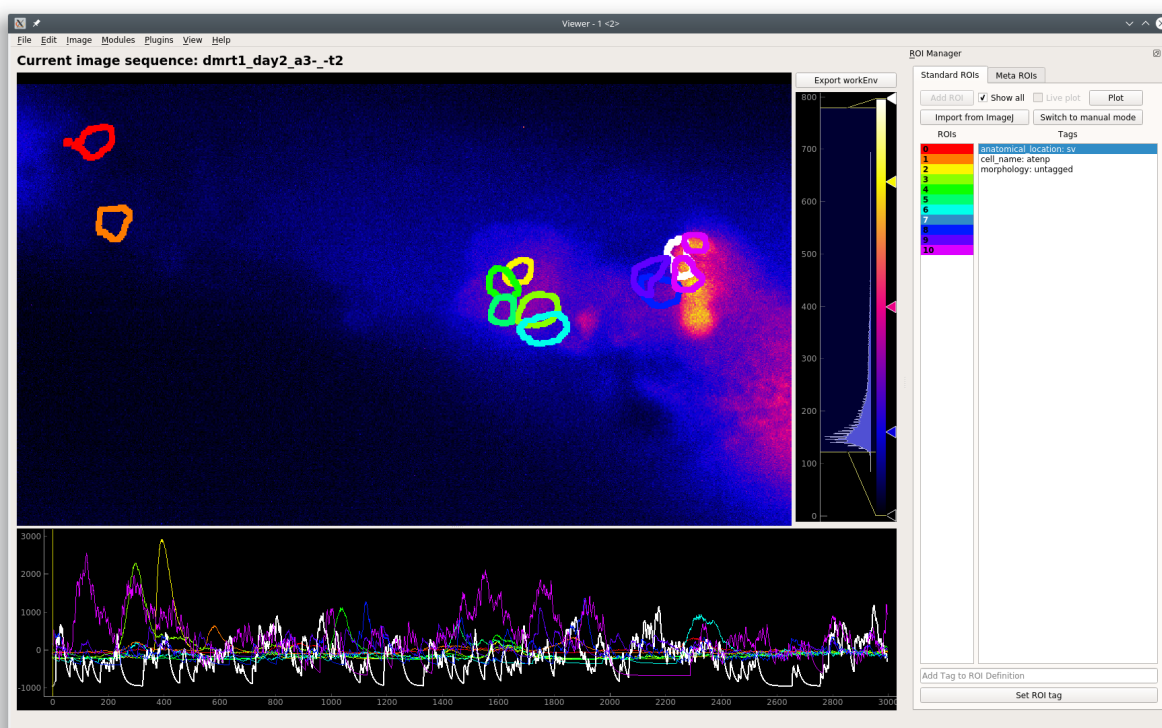
1 import pandas as pd
2 from mesmerize.plotting.utils import get_colormap
3
4 # Load dataframe from CSV
5 df = pd.read_csv('/share/data/longterm/4/kushal/crcns_datasets/pvc-7/122008_140124_
  ↳windowmix/stimulus.csv')
6
7 # Sort according to time (stimulus "start" frame column)
8 df.sort_values(by='start').reset_index(drop=True, 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.

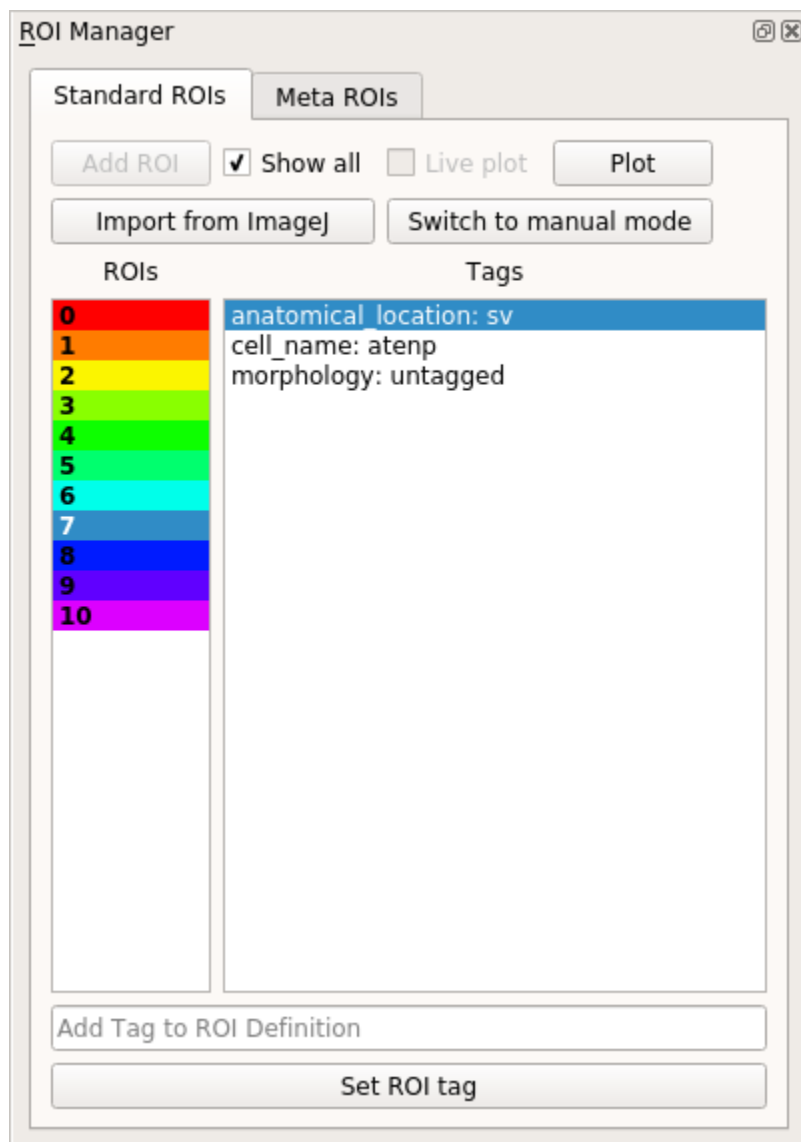
See also:

CNMF, *CNMF 3D*, and *CNMF E*.

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/>

12.1 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

Note: It is generally advisable to keep your ROI tags short with lowercase letters. When sharing your project you can provide a mapping for all your keys. This helps maintain consistency throughout your project and makes the data more readable.

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

12.2 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.

12.3 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.ManagerCNMFROI 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

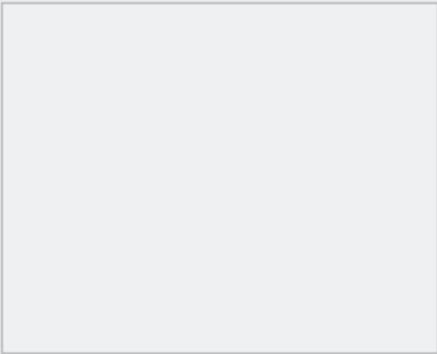
Perform motion correction using the NoRMCorre implementation in the CaImAn library.

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

CaImAn Motion Correction

☐ Use motion correction kwargs



Rigid correction

Output bit depth: Do not convert ▾

gSig_filt: 0 ▴ ▾

max shifts X (pixels): 0 ▴ ▾

max shifts Y (pixels): 0 ▴ ▾

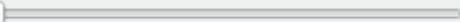
iterations for rigid: 1 ▴ ▾

If do only rigid correction add here

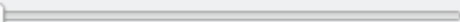
Elastic correction

max deviation from rigid: 3 ▴ ▾

strides (pixels): 0 ▴ ▾

☐ 

overlaps (pixels): 0 ▴ ▾

☐ 

upsample grid: 4 ▴ ▾

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.

You can also enter parameters as keyword arguments (kwargs) in the text box if you select "Use motion correction kwargs". This is useful if you want to enter parameters that cannot be entered in the GUI for example. **Use single quotes if you want to enter string kwargs, do not use double quotes.**

13.1 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.

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.

13.2 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

13.2.1 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
8 mc_kwargs = \
9 {
10     "max_shifts":      (6, 6)
11     "niter_rig":       2,
12     "max_dev":         3,
```

(continues on next page)

(continued from previous page)

```

13     "strides":          196,
14     "overlaps":        98,
15     "upsample_factor_grid": 4,
16     "gSig_filt"        (10, 10)  # Set to `None` for 2p data
17 }
18
19 params = \
20 {
21     'mc_kwargs':        mc_kwargs,  # the kwargs we set above
22     'item_name':        "will set later per file",
23     'output_bit_depth': "Do not convert"  # can also set to `8` or `16` if you want,
    ↳ the output in `8` or `16` bit
24 }
25
26 # Path to the dir containing images
27 files = glob("/full_path_to_raw_images/*.tiff")
28 # Sort in alphabetical order (should also work for numbers)
29 files.sort()
30
31 # Open each file, crop, and add to batch with 3 diff mot cor params
32 for i, path in enumerate(files):
33     print("Working on file " + str(i + 1) + " / " + str(len(files)))
34
35     # get json file path for the meta data
36     meta_path = path[:-5] + ".json"
37
38     # Create a new work environment with this image sequence
39     work_env = ViewerWorkEnv.from_tiff(path, "asarray-multi", meta_path)
40
41     # set it as the current work environment
42     vi.viewer.workEnv = work_env
43     vi.update_workEnv()
44
45     # Get caiman motion correction module, hide=False to not show GUI
46     mc_module = get_module("caiman_motion_correction", hide=True)
47
48     # Set name for this video file
49     name = os.path.basename(path)[:-5]
50     params["item_name"] = name
51
52     # First variant of params
53     params["mc_kwargs"]["strides"] = 196
54     params["mc_kwargs"]["overlaps"] = 98
55
56     # Add one variant of params for this video to the batch
57     mc_module.add_to_batch(params)
58
59     # Try another variant of params
60     params["mc_kwargs"]["strides"] = 256
61     params["mc_kwargs"]["overlaps"] = 128
62
63     # Set these params and add to batch
64     mc_module.add_to_batch(params)
65
66     # Try one more variant of params
67     params["mc_kwargs"]["strides"] = 296
68     params["mc_kwargs"]["overlaps"] = 148

```

(continues on next page)

(continued from previous page)

```

69
70     # Set these params and add to batch
71     mc_module.add_to_batch(params)
72
73     # If you want to process the batch after adding the items uncomment the following_
74     ↪ lines
75     #bm = get_batch_manager()
76     #bm.process_batch(clear_viewers=True)

```

13.2.2 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
12 ↪ " (See below)
13 crop_params = \
14 {
15     "projection":      "max+std",
16     "method":          "threshold",
17     "denoise_params":  (32, 32),
18 }
19
20 # Spectral saliency is another method
21 # You can try and play around with the parameters
22 # If the cropping is insufficient, you can set "projection" to just "max" or "std"
23 # If you get too much junk blackness around the animal try increasing denoise_params
24 # or reduce padding. Default padding is 30 (when nothing is specified like above)
25 crop_params_salient = \
26 {
27     "projection":      "max+std",
28     "method":          "spectral_saliency",
29     "denoise_params":  (16, 16),
30     "padding":         40
31 }
32
33 # Motion correction params.
34 mc_kwargs = \
35 {
36     "max_shifts":      (6, 6)

```

(continues on next page)

(continued from previous page)

```

36     "niter_rig":                2,
37     "max_dev":                  3,
38     "strides":                  196,
39     "overlaps":                  98,
40     "upsample_factor_grid": 4,
41     "gSig_filt"                 (10, 10)  # Set to `None` for 2p data
42 }
43
44 params = \
45 {
46     'mc_kwargs':                mc_kwargs, # the kwargs we set above
47     'item_name':                 "will set later per file",
48     'output_bit_depth': "Do not convert" # can also set to `8` or `16` if you want
↳ the output in `8` or `16` bit
49 }
50
51 # Path to the dir containing images
52 files = glob("/full_path_to_raw_images/*.tiff")
53 # Sort in alphabetical order (should also work for numbers)
54 files.sort()
55
56 # Open each file, crop, and add to batch with 3 diff mot cor params
57 for i, path in enumerate(files):
58     print("Working on file " + str(i + 1) + " / " + str(len(files)))
59
60     # get json file path for the meta data
61     meta_path = path[:-5] + ".json"
62
63     # Create a new work environment with this image sequence
64     work_env = ViewerWorkEnv.from_tiff(path, "asarray-multi", meta_path)
65
66     # autocrope the image sequence in the work environment
67     raw_seq = work_env.imgdata.seq
68     # Auto crop the image sequence
69     print("Cropping file: " + str(i + 1))
70
71     cropped = auto_crop.crop(raw_seq, crop_params)
72     # Set work env img seq to the cropped one and update
73     work_env.imgdata.seq = cropped
74
75     # update the work environment
76     vi.viewer.workEnv = work_env
77     vi.update_workEnv()
78
79     # Get caiman motion correction module, hide=False to not show GUI
80     mc_module = get_module("caiman_motion_correction", hide=True)
81
82     # Set name for this video file
83     name = os.path.basename(path)[:-5]
84     params["item_name"] = name
85
86     # First variant of params
87     params["mc_kwargs"]["strides"] = 196
88     params["mc_kwargs"]["overlaps"] = 98
89
90     # Add one variant of params for this video to the batch
91     mc_module.add_to_batch(params)

```

(continues on next page)

(continued from previous page)

```
92
93     # Try another variant of params
94     params["mc_kwargs"]["strides"] = 256
95     params["mc_kwargs"]["overlaps"] = 128
96
97     # Set these params and add to batch
98     mc_module.add_to_batch(params)
99
100    # Try one more variant of params
101    params["mc_kwargs"]["strides"] = 296
102    params["mc_kwargs"]["overlaps"] = 148
103
104    # Set these params and add to batch
105    mc_module.add_to_batch(params)
106
107    # If you want to process the batch after adding the items uncomment the following_
    ↪ lines
108    #bm = get_batch_manager()
109    #bm.process_batch(clear_viewers=True)
```


CNMF

Perform CNMF 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

Order of the autoregressive system

p: 2

Global number of background components

gnb: 1

Merging threshold, max correlation allowed

merge_thresh: 0.70

Half size of patch in pixels

rf: 50

amount of overlap between the patches in pixels

stride_cnmf: 30

Number of neurons/cell per patch

k: 10

Expected half size of neurons (x, y)

gSig: 5 5

ssub: 1 tsub: 1

method_init greedy_roi

Ain:

☐ Use CNMF kwargs

Signal to noise ratio for accepting a component

min_SNR: 2.50

Space correlation threshold for accepting a component

rval_thr: 0.80

Threshold for CNN based classifier

cnn_thr: 0.80

cnn_lowest

0.10

Average decay time of calcium spikes (seconds)

decay_time: 1.00

☐ Use evaluation params

perform second iteration of cnmf by re-fitting the components

refit ☒

Enter name

Add to batch

Parameters

Please see the CaImAn demo notebook mentioned above to understand the parameters. The Caiman docs also provide descriptions of the parameters: <https://caiman.readthedocs.io/>

You can also enter parameters for CNMF and component evaluation as keyword arguments (kwargs) in the the respective text boxes if you select “Use CNMF kwargs” or “Use evaluation params”. This is useful if you want to enter parameters that cannot be entered in the GUI for example. **Use single quotes if you want to enter string kwargs, do not use double quotes.**

Note: The parameters used for CNMF 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 CNMF in the viewer, execute `get_workEnv().history_trace` in the viewer console and look for the ‘cnmf’ entry.

14.1 Script usage

A script can be used to add CNMF 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

```

1 def reset_params():
2     # CNMF Params that we will use for each item
3     cnmf_kwargs = \
4     {
5         'p': 2,
6         'gnb': 1,
7         'merge_thresh': 0.25,
8         'rf': 70,
9         'stride': 40,
10        'k': 16,
11        'gSig': (8, 8),
12        'gSiz': (33, 33)
13    }
14
15    # component evaluation params
16    eval_kwargs = \
17    {
18        'min_SNR': 2.5,
19        'rval_thr': 0.8,
20        'min_cnn_thr': 0.8,
21        'cnn_lowest': 0.1,
22        'decay_time': 2.0,
23    }
24
25    # the dict that will be passed to the mesmerize caiman module
26    params = \
27    {
28        "cnmf_kwargs": cnmf_kwargs,
29        "eval_kwargs": eval_kwargs,
30        "refit": True, # if you want to perform a refit
31        "item_name": "will set later per file",
32    }
33
34    return params
35
36    # Get the batch manager
37    bm = get_batch_manager()
38    cnmf_mod = get_module('cnmf', hide=True)
39
40    # Start index if we want to start processing the new items after they have been added
41    start_ix = bm.df.index.size + 1
42
43    # This example uses motion corrected output items from the batch manager
44    # You can also open image files directly from disk, see the motion correction
45    # script examples to see how to open images from disk.
46    for ix, r in bm.df.iterrows():
47        # Use output of items 6 - 12
48        # for example if items 6 - 12 were motion correction items

```

(continues on next page)

(continued from previous page)

```

49     if ix < 6:
50         continue
51     if ix > 12: # You need to set a break point, else the batch grows infinitely
52         break
53
54     # get the first variant of params
55     params = reset_parms()
56
57     # Get the name of the mot cor item
58     name = r['name']
59
60     # Set the name for the new cnmf item
61     params['item_name'] = name
62
63     # Load the mot cor output
64     bm.load_item_output(module='caiman_motion_correction', viewers=viewer, UUID=r[
65 ↪ 'uuid'])
66
67     # Set the sampling rate of the data
68     params['eval_kwargs']['fr'] = vi.viewer.workEnv.imgdata.meta['fps']
69
70     # Get the border_pix value from the motion correction output
71     # skip this if loading files that don't have NaNs on the image borders
72     history_trace = vi.viewer.workEnv.history_trace
73     border_pix = next(d for ix, d in enumerate(history_trace) if 'caiman_motion_
74 ↪ correction' in d) ['caiman_motion_correction'] ['bord_px']
75
76     # Set the border_pix values
77     params['border_pix'] = border_pix
78     params['cnmf_kwargs']['border_pix'] = border_pix
79
80     # Add to batch
81     cnmf_mod.add_to_batch(params)
82
83     # change some of the params and add this variant to batch
84     params['cnmf_kwargs']['gSig'] = (10, 10)
85     params['cnmf_kwargs']['gSiz'] = (41, 41)
86
87     # Add to batch with this params variant
88     cnmf_mod.add_to_batch(params)
89
90     # another parameter variant
91     params['eval_kwargs']['rval_thr'] = 0.7
92     params['eval_kwargs']['min_cnn_thr'] = 0.65
93
94     # Add to batch with this params variant
95     cnmf_mod.add_to_batch(params)
96
97     # Cleanup the work environment
98     vi._clear_workEnv()
99
100     # Uncomment the last two lines to start the batch as well
101     #bm.process_batch(start_ix, clear_viewers=True)

```


CNMF 3D

Perform 3D CNMF 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.** https://github.com/flatironinstitute/CaImAn/blob/master/demos/notebooks/demo_caiman_cnmf_3D.ipynb

The screenshot shows the 'CNMF 3D' window with the following parameters and controls:

- p**: A numeric input field set to 2.
- rval threshold**: A numeric input field set to 0.70.
- ☒ **Use patches**: A checked checkbox.
- rf**: A numeric input field set to 25.
- stride**: A numeric input field set to 15.
- min SNR**: A numeric input field set to 2.50.
- decay time**: A numeric input field set to 2.00.
- ☐ **Use evaluation params**: An unchecked checkbox.
- k**: A numeric input field set to 5.
- gSig, (x, y z)**: Three separate numeric input fields, each set to 3.
- merge threshold**: A numeric input field set to 0.80.
- ☐ **Use CNMF kwargs**: An unchecked checkbox.
- ☐ **refit**: An unchecked checkbox.
- ☐ **Keep memmap of this batch item**: An unchecked checkbox.
- ☐ **Use memmap from previous batch item**: An unchecked checkbox.
- Enter UUID**: A text input field.
- Enter name**: A text input field.
- Add to batch**: A button.

Parameters

Please see the CaImAn demo notebook mentioned above to understand the parameters. The Caiman docs also provide descriptions of the parameters: <https://caiman.readthedocs.io/>

You can also enter parameters for CNMF and component evaluation as keyword arguments (kwargs) in the respective text boxes if you select “Use CNMF kwargs” or “Use evaluation params”. This is useful if you want to enter parameters that cannot be entered in the GUI for example. **Use single quotes if you want to enter string kwargs, do not use double quotes.**

Note: The parameters used for 3D CNMF 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 3D CNMF in the viewer, execute

`get_workEnv().history_trace` in the viewer console and look for the 'cnmf_3d' entry.

15.1 Script Usage

A script can be used to add CNMF batch items. This is much faster than using the GUI. This example sets the work environment from the output of a batch item.

See also:

Script Editor

This example loads 3D sequences from disk & adds them to a batch with 3 parameter variants.

```

1  # just so we can reset the params for each new image file
2  def reset_params():
3      # CNMF Params that we will use for each item
4      cnmf_kwargs = \
5          {
6              'p': 2,
7              'merge_thresh': 0.8,
8              'k': 50,
9              'gSig': (10, 10, 1),
10             'gSiz': (41, 41, 4)
11         }
12
13     # component evaluation params
14     eval_kwargs = \
15         {
16             'min_SNR': 3.0,
17             'rval_thr': 0.75,
18             'decay_time': 1.0,
19         }
20
21     # the dict that will be passed to the mesmerize caiman module
22     params = \
23         {
24             "cnmf_kwargs": cnmf_kwargs,
25             "eval_kwargs": eval_kwargs,
26             "refit": True, # if you want to perform a refit
27             "item_name": "will set later per file",
28             "use_patches": False,
29             "use_memmap": False, # re-use the memmap from a previous batch item,
↪ reduces computation time
30             "memmap_uuid": None, # UUID (as a str) of the batch item to use the memmap,
↪ from
31             "keep_memmmmap": False # keep the memmap of this batch item
32         }
33
34     return params
35
36 # get the 3d cnmf module
37 cnmf_mod = get_module('cnmf_3d', hide=True)
38
39 # Path to the dir containing images
40 files = glob("/full_path_to_raw_images/*.tiff")

```

(continues on next page)

(continued from previous page)

```

42 # Sort in alphabetical order (should also work for numbers)
43 files.sort()
44
45 # Open each file, crop, and add to batch with 3 diff mot cor params
46 for i, path in enumerate(files):
47     print("Working on file " + str(i + 1) + " / " + str(len(files)))
48
49     # get json file path for the meta data
50     meta_path = path[:-5] + ".json"
51
52     # Create a new work environment with this image sequence
53     vi.viewer.workEnv = ViewerWorkEnv.from_tiff(path=path,          # tiff file path
54                                                method='imread',    # use imread
55                                                meta_path=meta_path, # json metadata
56     ↪file path
57                                                axes_order=None)      # default axes
58     ↪order
59                                                # see Mesmerize
60     ↪Tiff file module docs for more info on axes order
61
62     # update the work environment
63     vi.update_workEnv()
64
65     # get the first variant of params
66     params = reset_params()
67
68     # Set name for this video file
69     name = os.path.basename(path)[:-5]
70     params["item_name"] = name
71
72     # add batch item with one variant of params
73     u = cnmf_mod.add_to_batch(params)
74
75     # add the same image but change some params
76     params["cnmf_kwargs"]["gSig"] = (12, 12, 1)
77     params["eval_kwargs"]["min_SNR"] = 2.5
78
79     # use the same memmap as the previous batch item
80     # since it's the same image
81     params["use_memmap"] = True
82     params["memmap_uuid"] = str(u)
83
84     # add this param variant to the batch
85     cnmf_mod.add_to_batch(params)
86
87     # one more variant of params
88     params["eval_kwargs"]["min_SNR"] = 2.0
89
90     # add this param variant to the batch
91     cnmf_mod.add_to_batch(params)

```

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

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: 10

Stop here:

Add to batch

CNMF-E

Ain:

p: 1

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

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

ssub: 1

tsub: 1

☐ low_rank_background

ring_size_factor: 1.50

deconvolution: oasis

merge_thresh: 0.70

☐ Use CNMF_kwargs

Evaluation params

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

☐ Use evaluation params

Perform CNMF-E:

Add to batch

Parameters

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

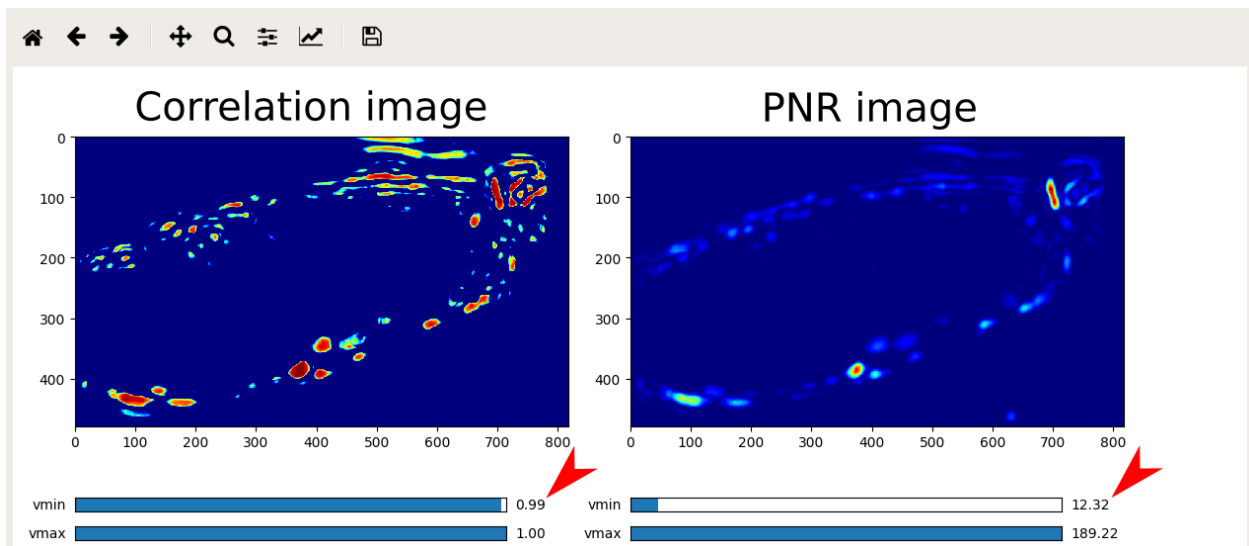
Please see the CaImAn demo notebook mentioned above to understand the rest of the parameters. The Caiman docs also provide descriptions of the parameters: <https://caiman.readthedocs.io/>

You can also enter parameters for CNMF and component evaluation as keyword arguments (kwargs) in the the respective text boxes if you select “Use CNMF kwags” or “Use evaluation params”. This is useful if you want to enter parameters that cannot be entered in the GUI for example. **Use single quotes if you want to enter string kwargs, do not use double quotes.**

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 module 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     # Load the output of the motion corrected batch item
15     # The output will load into the viewer that this script
16     # is running in.
17     bm.load_item_output(module='caiman_motion_correction', viewers=viewer, UUID=r[
18 ↪ 'uuid'])
19
20     # Get the currently set params
21     # You just need the dict with all the correct keys
22     # You will just modify the "gSig" and "item_name" keys
23     params = cnmfe_mod.get_params(item_type='corr_pnr', group_params=True)
24
25     # Get the name of the mot cor item
26     name = r['name']
27     params['item_name'] = name
28
29     params['border_pix'] = border_pix
30
31     # Set the gSig and name params
32     params['corr_pnr_kwargs']['gSig'] = 8
33
34     # Add to batch
35     cnmfe_mod.add_to_batch_corr_pnr(params)

```

(continues on next page)

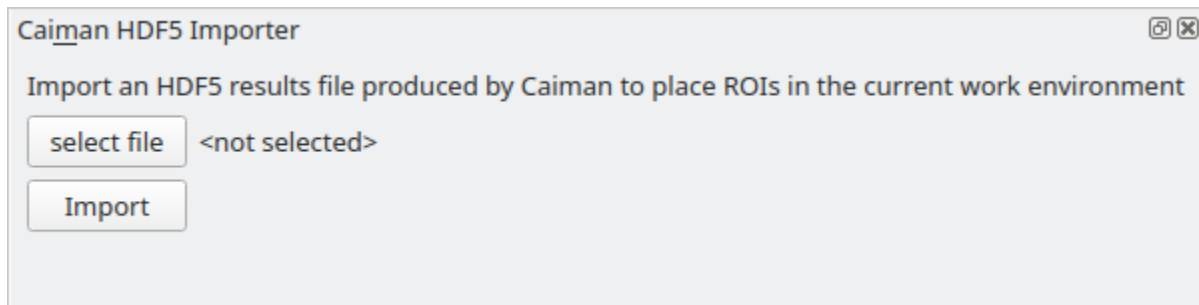
(continued from previous page)

```
36     # Another variant of params
37     params['corr_pnr_kwargs']['gSig'] = 10
38
39     # Add to batch with this variant of params
40     cnmfe_mod.add_to_batch_corr_pnr(params)
41
42     # Cleanup the work environment
43     vi._clear_workEnv()
44
45     # Start the batch from the start_ix
46     bm.process_batch(start_ix, clear_viewers=True)
```

16.2.2 CNMFE

CAIMAN HDF5 IMPORTER

You can import HDF5 files containing CNMF results that were produced externally by Caiman. The ROIs produced by CNMF, 3D-CNMF or CNMFE will be imported into the current *work environment* and placed onto the image that is currently open.



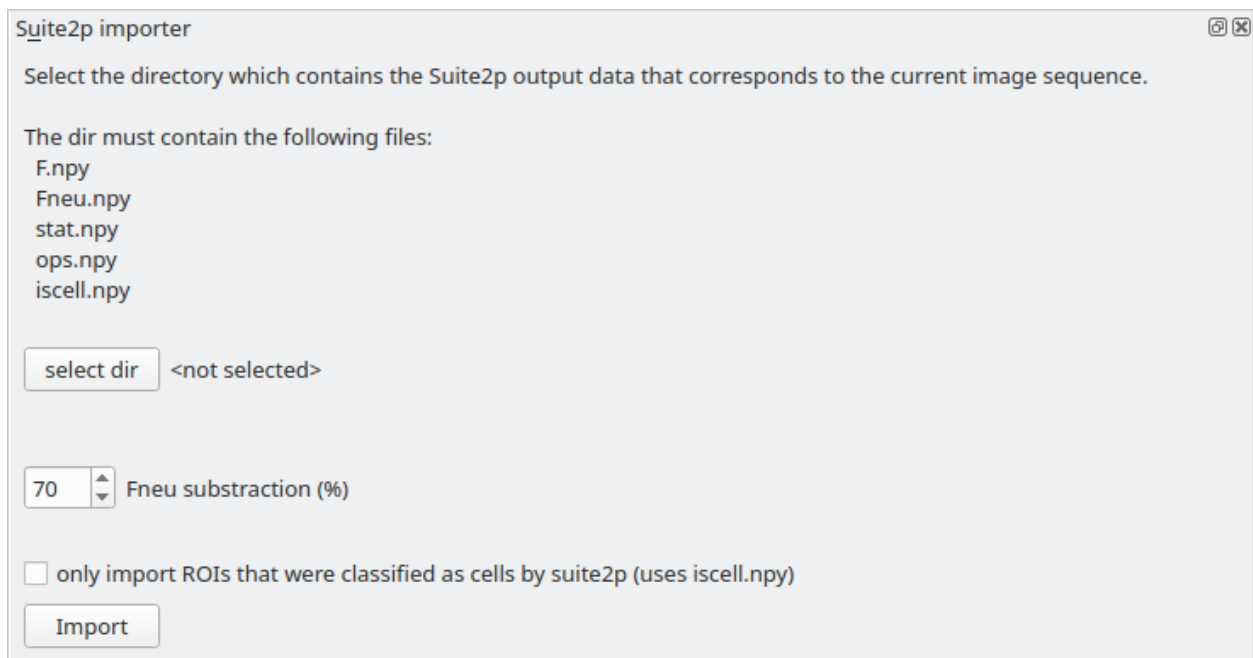
You can also use this module through the *viewer console*, or in the *Script Editor* instead of clicking buttons.

Example

```
1 # get the module, hide the GUI
2 caiman_importer = get_module('caiman_importer', hide=True)
3
4 # import the file
5 caiman_importer.import_file('/path/to/file.hdf5')
```


SUITE2P IMPORTER

You can load Suite2p output files to import ROIs into the current *work environment*. This places the Suite2p-derived ROIs onto the image that is currently open.



The screenshot shows a window titled "Suite2p importer" with a close button in the top right corner. The window contains the following elements:

- A text instruction: "Select the directory which contains the Suite2p output data that corresponds to the current image sequence."
- A list of required files: "The dir must contain the following files:" followed by "F.npy", "Fneu.npy", "stat.npy", "ops.npy", and "iscell.npy".
- A "select dir" button next to a text field displaying "<not selected>".
- A spin box set to "70" next to the label "Fneu subtraction (%)".
- An unchecked checkbox with the label "only import ROIs that were classified as cells by suite2p (uses iscell.npy)".
- An "Import" button at the bottom.

You can also use this module through the *viewer console*, or in the *Script Editor* instead of clicking buttons.

Example

```
1 # get the module, hide the GUI
2 s2p_importer = get_module('suite2p_importer', hide=True)
3
4 # set the path to the dir containing the suite2p output files
5 s2p_importer.data.set_dir('/path/to/dir')
6
7 # set the amount of neuropil contamination to subtract
8 s2p_importer.data.Fneu_sub = 0.7
9
10 # import the suite2p data into the current work environment
11 s2p_importer.import_rois()
12
13 # clear the data from the importer before importing another directory
```

(continues on next page)

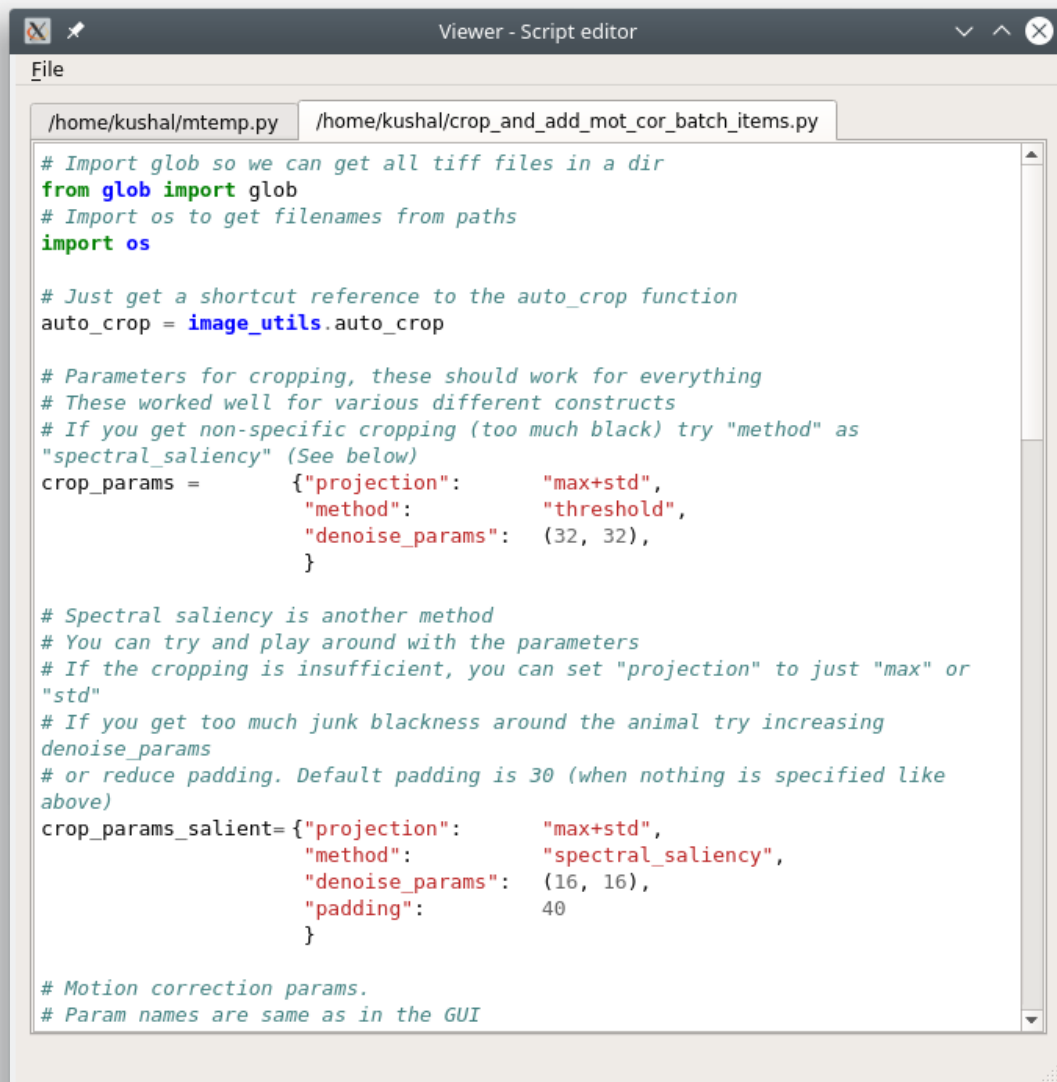
(continued from previous page)

```
14 # this doesn't do anything to the viewer work environment, just clears the importer_
    ↪data
15 s2p_importer.data.clear()
```

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](#)

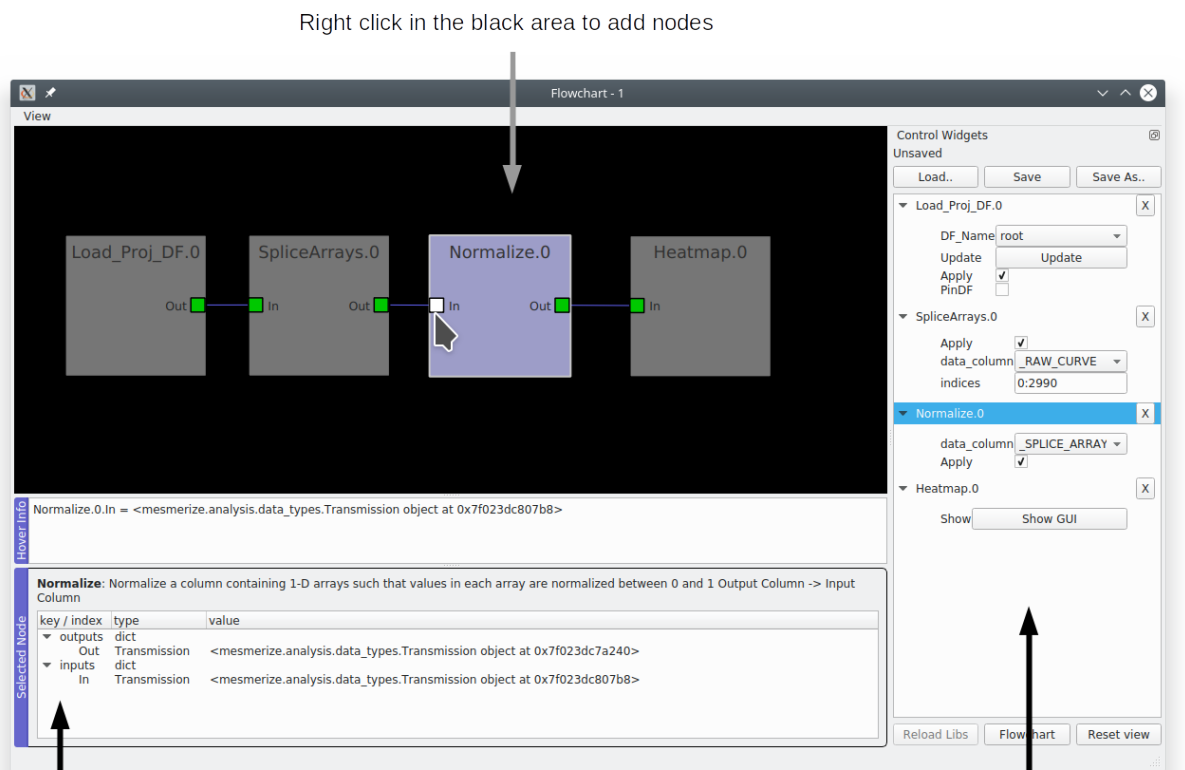
Warning: There is no auto-save function

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.

20.1 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 *Project DataFrame* (or sub-DataFrame) by the *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.

20.2 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:

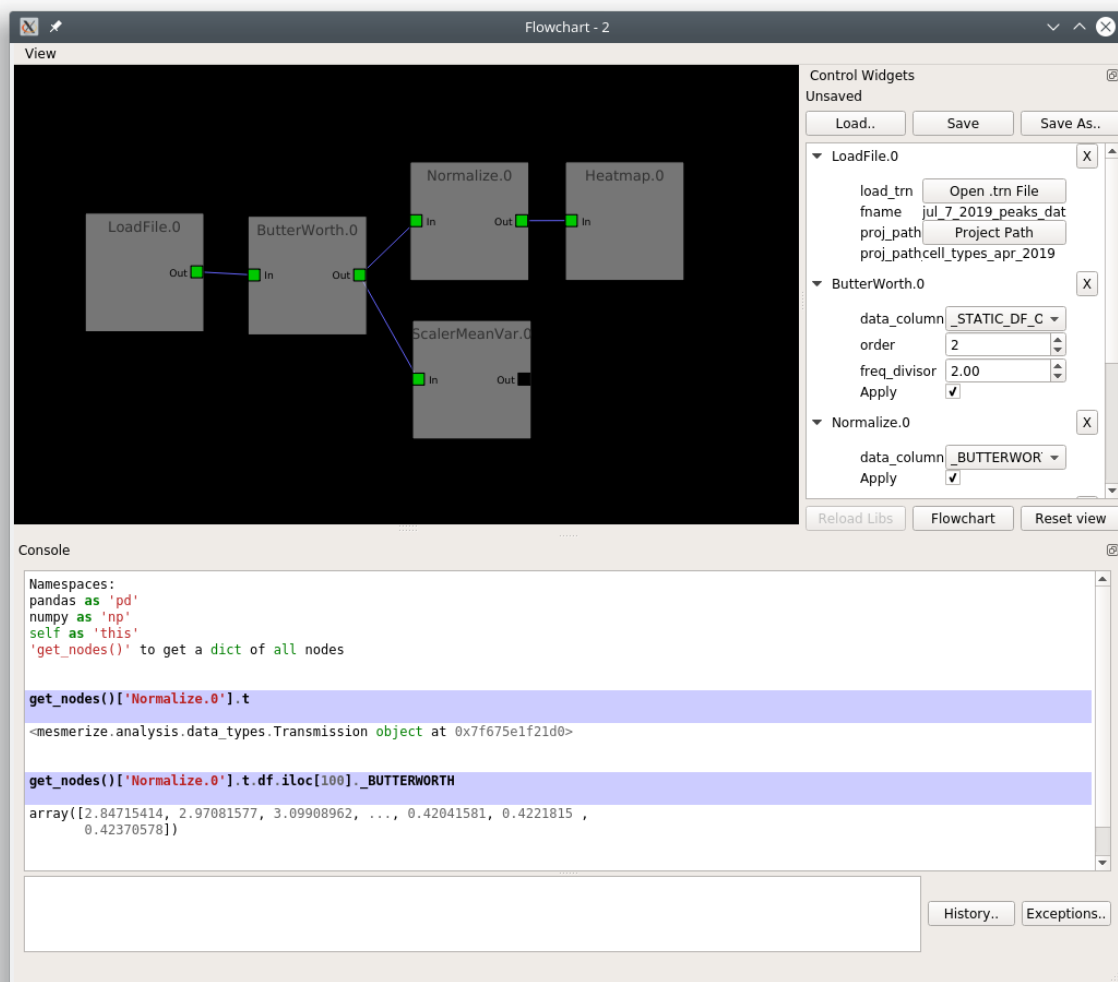
If you are unfamiliar with the console see the overview on [Consoles](#)

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



20.3 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*.

20.3.1 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]
```

20.3.2 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}}

```


21.1 Data

These nodes are for performing general data related operations

21.1.1 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.

21.1.2 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.

21.1.3 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.

21.1.4 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

21.1.5 ViewTransmission

Source

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

21.1.6 ViewHistory

Source

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

21.1.7 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

21.1.8 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

21.1.9 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

21.1.10 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.

21.2 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>
```

21.2.1 BeeswarmPlots

Source

Based on ppytgraph Beeswarm plots.

Visualize data points as a pseudoscatter 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

21.2.2 Heatmap

Source

Used for visualizing numerical arrays in the form of a heatmap. Also used for visualizing a hierarchical 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.

21.2.3 CrossCorr

Source

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

21.2.4 Plot

Source

For information on the plot widget see *Simple Plot*

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

21.2.5 Proportions

Source

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

For information on the plot widget see *Proportions Plot*

21.2.6 ScatterPlot

Source

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

For information on the plot widget see *Scatter Plot*

21.3 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>

21.3.1 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

21.3.2 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
<code>data_column</code>	Data column containing numerical arrays to be filtered
<code>win-dow_length</code>	Size of windows for fitting the polynomials. Must be an odd number.
<code>polyorder</code>	Order of polynomials to fit into the windows. Must be less than <code>win-dow_length</code>
Apply	Process data through this node

21.3.3 PowSpecDens

21.3.4 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
<code>data_column</code>	Data column containing numerical arrays to be resampled
<code>Rs</code>	New sampling rate in Tu units of time.
<code>Tu</code>	Time unit
Apply	Process data through this node

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

21.3.5 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
<code>data_column</code>	Data column containing numerical arrays to be scaled
<code>mu</code>	Mean of the output time series
<code>std</code>	Standard Deviation of the output time series
Apply	Process data through this node

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

21.3.6 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

21.3.7 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

21.3.8 iRFFT

Source

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

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

21.3.9 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

21.3.10 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

21.4 Math

Nodes for performing basic Math functions

21.4.1 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

21.4.2 TVDiff

Source

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

21.4.3 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

21.4.4 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

21.4.5 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

21.4.6 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

21.4.7 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

21.4.8 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

21.4.9 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`

21.5 Biology

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

21.5.1 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>

21.5.2 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`

21.5.3 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

21.6 Clustering

21.6.1 KShape

Source

Perform KShape clustering. For more information see *KShape plot*.

21.6.2 KMeans

Source

Basically `sklearn.cluster.KMeans`.

21.7 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.

21.7.1 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

21.7.2 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*.

21.7.3 Inconsistent

21.7.4 MaxIncStat

21.7.5 MaxInconsistent

21.8 Transform

Nodes for transforming data

21.8.1 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>

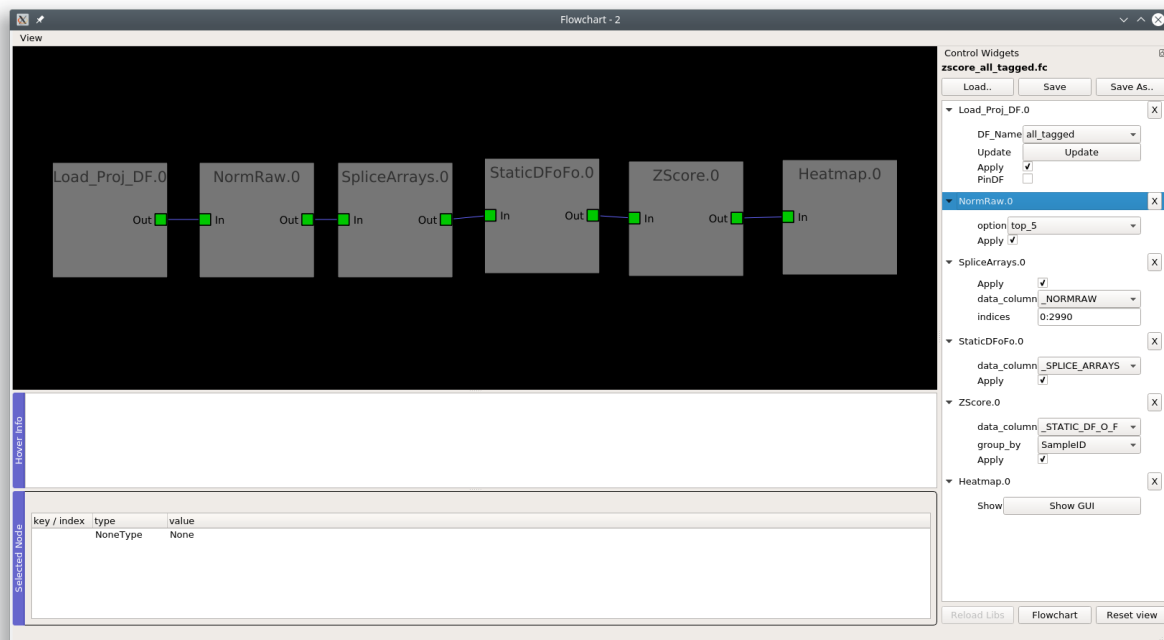
C. elegans dataset: <https://doi.org/10.6084/m9.figshare.10287113>

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

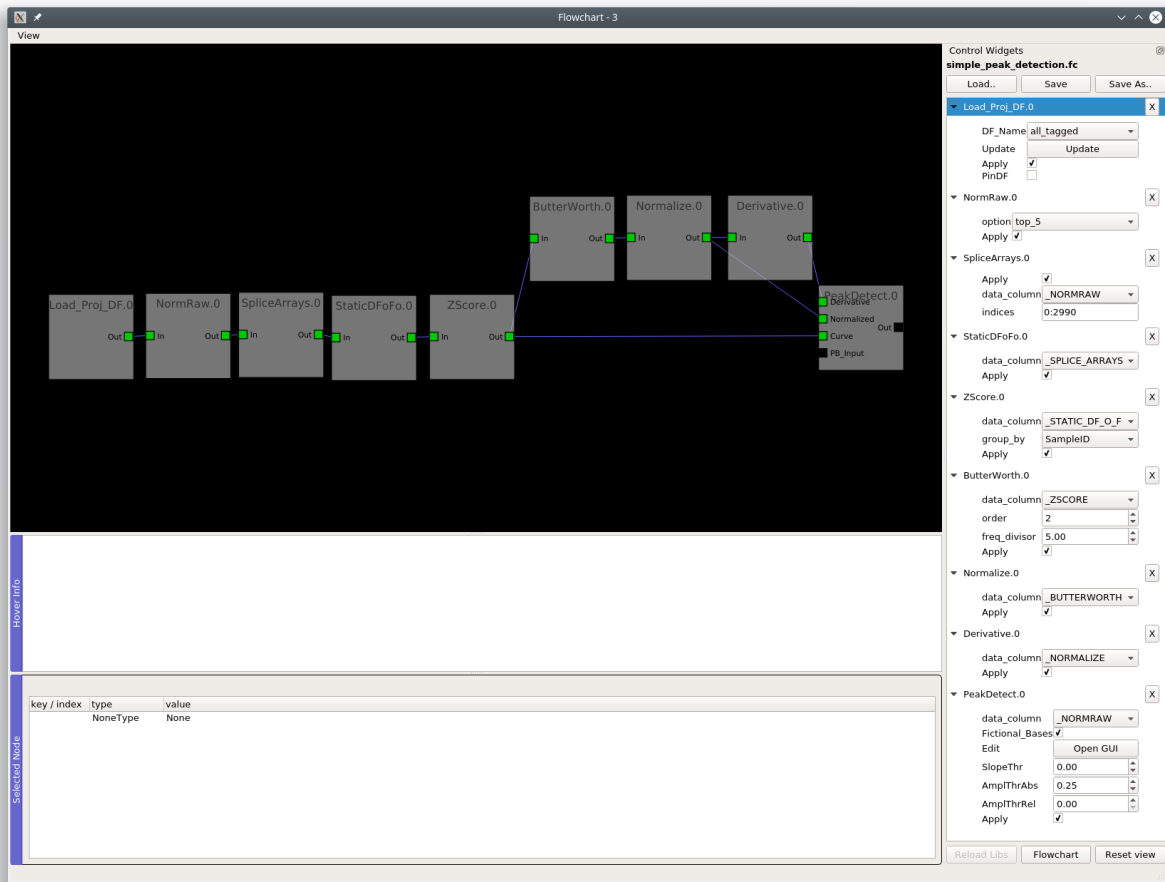
22.1 Screenshots

Flowchart screenshots from the C. intestinalis dataset.

22.1.1 Z-score



22.1.2 Peak detection



22.1.3 Hierarchical clustering

The screenshot displays the Mesmerize Flowchart interface for a hierarchical clustering workflow. The main canvas shows a sequence of processing nodes: Load_Proj_DF.0, SpliceArrays.0, Normalize.0, RFFT.0, AbsoluteValue.0, LogTransform.0, SpliceArrays.1, Linkage.0, and FCluster.0. The FCluster.0 node is highlighted with a red border. Below the canvas, a 'Selected Node' panel shows a table with the following data:

key / index	type	value
	NoneType	None

On the right, the 'Control Widgets' panel for 'hierarchical_clustering_freq_domain.fc' provides configuration options for each node:

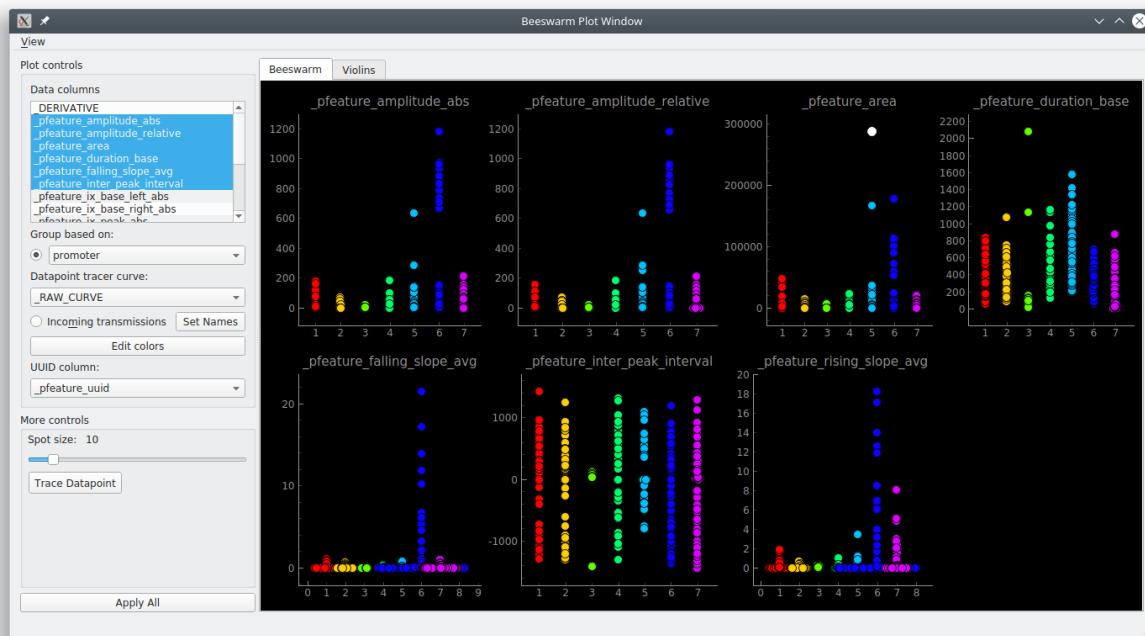
- Load_Proj_DF.0**: DF_Name: all_tagged, Update: [button], Apply: ☒, PinDF: ☐
- SpliceArrays.0**: Apply: ☒, data_column: RAW_CURVE, indices: 0:2990
- Normalize.0**: data_column: SPLICE_ARRAYS, Apply: ☒
- RFFT.0**: data_column: NORMALIZE, Apply: ☒
- AbsoluteValue.0**: data_column: RFFT, Apply: ☒
- LogTransform.0**: data_column: ABSOLUTE_VALUE, transform: log10, Apply: ☒
- SpliceArrays.1**: Apply: ☒, data_column: LOG_TRANSFORM, indices: 0:1000
- Linkage.0**: data_column: ABSOLUTE_VALUE, method: complete, metric: wasserstein, optimal_order: ☒, Apply: ☐
- FCluster.0**: threshold: 10.00, criterion: maxclust, depth: 1

At the bottom of the control panel are buttons for 'Reload Libs', 'Flowchart', and 'Reset view'.

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

23.1 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

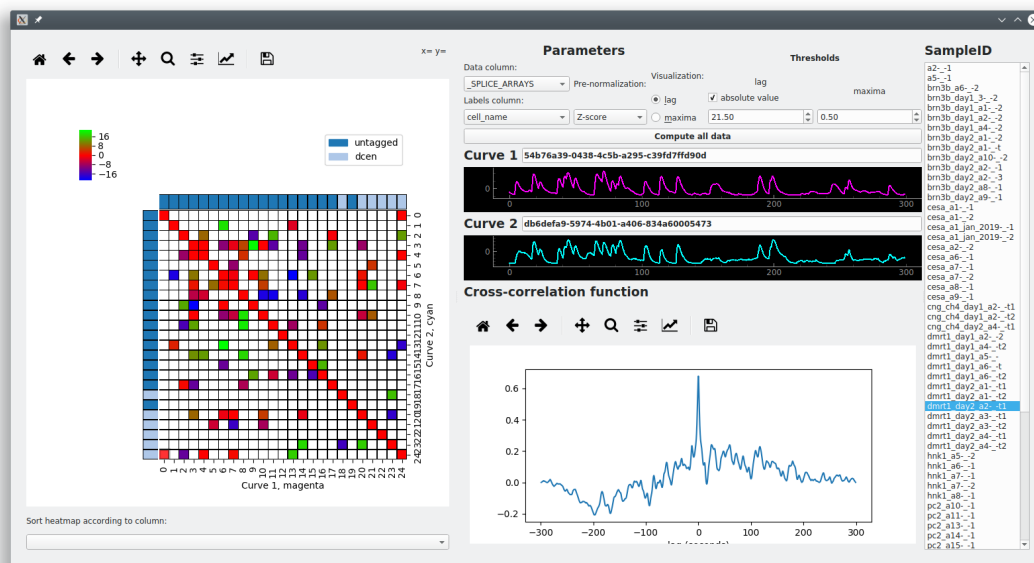
Currently the *Heatmap*, *Scatter*, *Proportions plot* and *KShape* follow a uniform structure allowing internal access to the data and plot axes. Refer to their *Base API*. For example, through their consoles you can access the *Transmission* containing data for the current plot, manually save the plot,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

25.1 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.

25.2 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.

25.3 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.

25.4 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.

25.4.1 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.

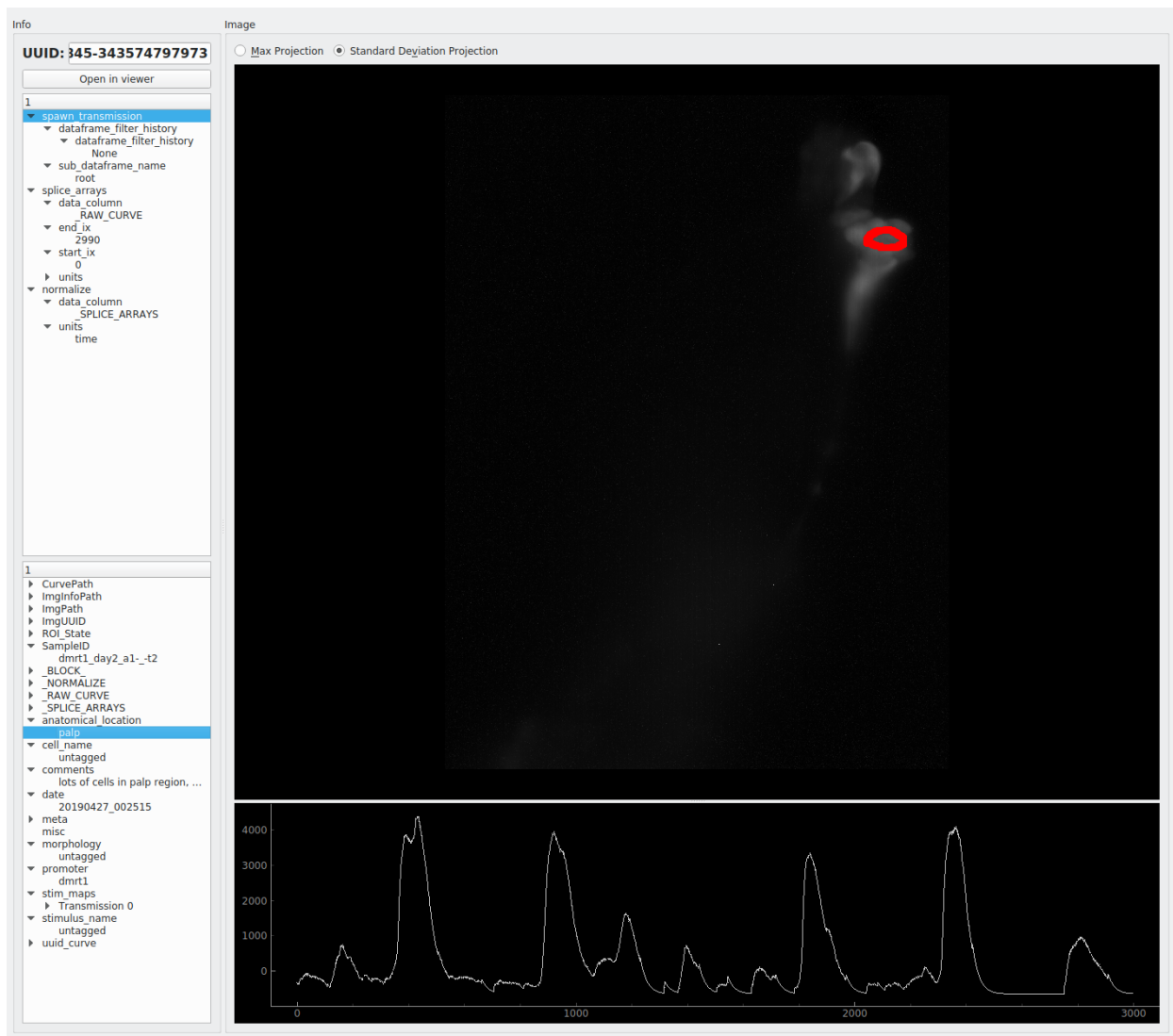
DATAPPOINT TRACER

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.

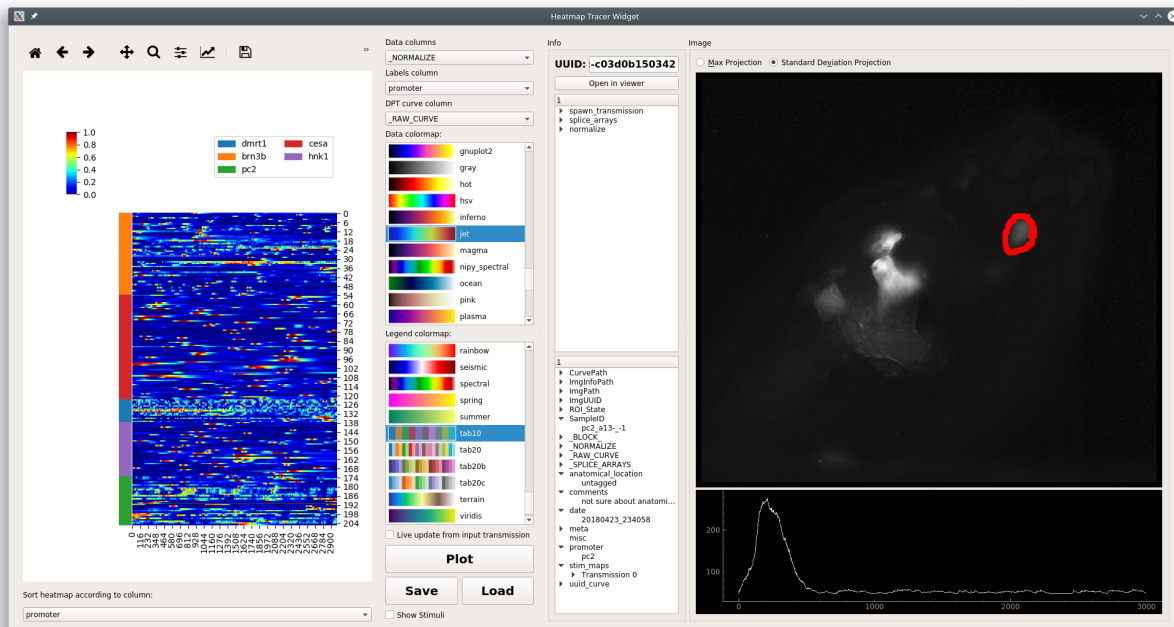
HEATMAP

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.

27.2 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

27.2.1 Namespace

reference	Description
this	The higher-level <i>HeatmapTracerWidget</i> instance, i.e. the entire widget
this.transmission	Current input <i>Transmission</i>
get_plot_area()	Returns the lower-level <i>Heatmap</i> 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

27.2.2 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.

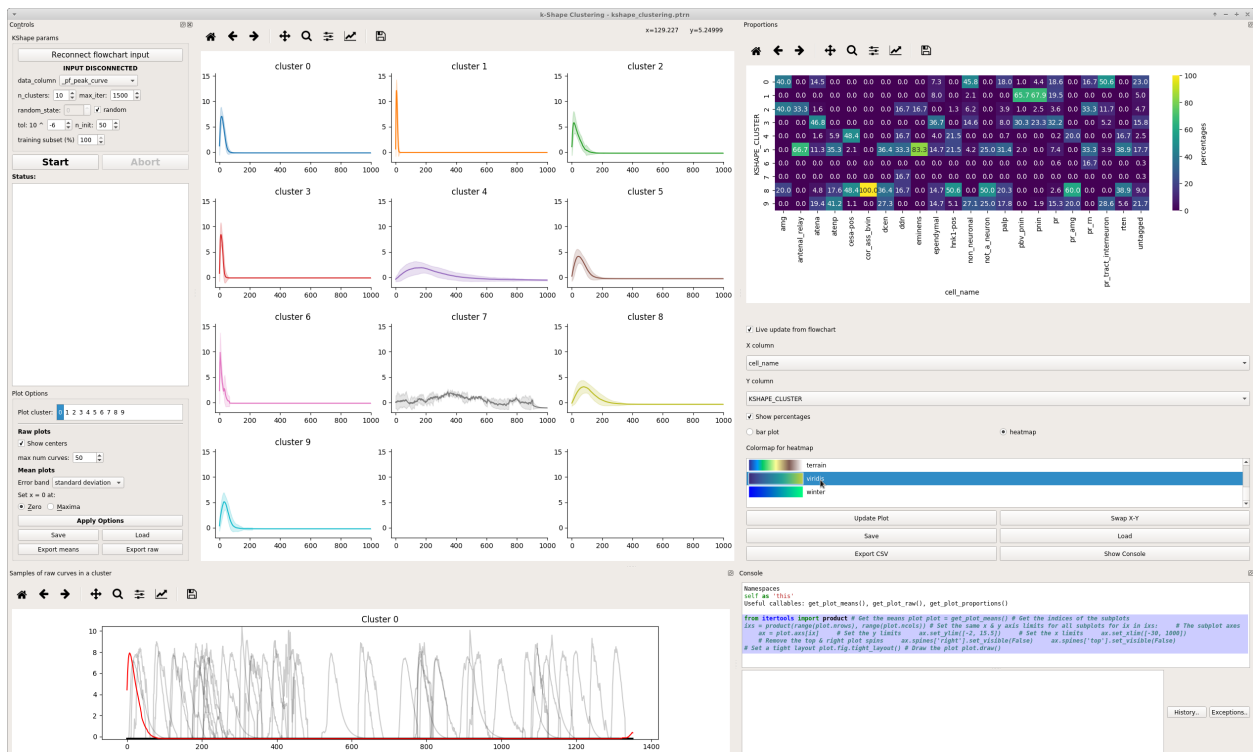
This GUI uses the `tslearn.clustering.KShape` implementation.

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

28.1 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.

28.2 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*

28.3 Console

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

See also:

API reference

28.3.1 Namespace

reference	Description
this	The higher-level <i>KShape</i> 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 <i>Proportions Widget</i>

28.3.2 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
15             ↪', pad_inches=0)
16
17 # Reset the figure size and draw
18 fig.set_size_inches(orig_size)
19 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!

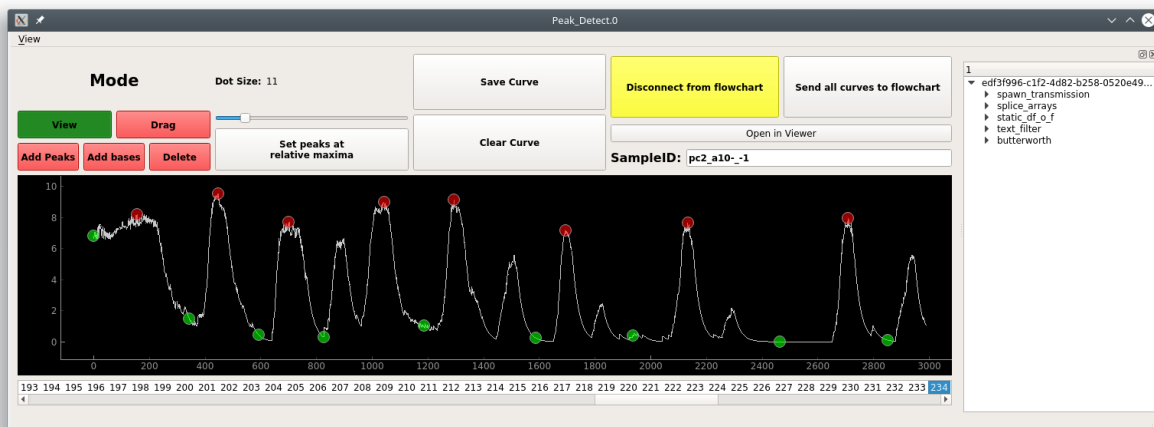
PEAK EDITOR

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

29.1 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.

29.1.1 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

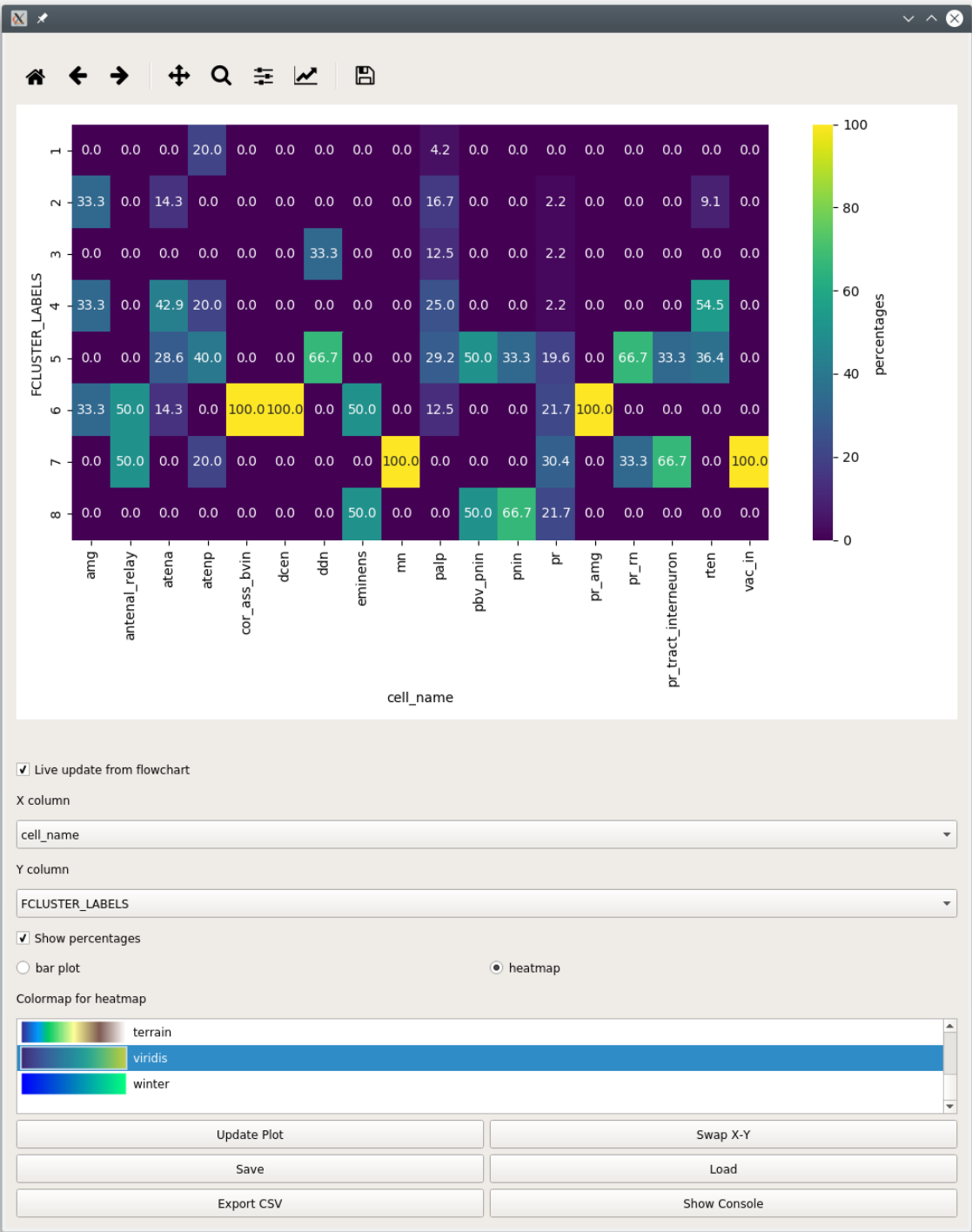
History Tree Widget

PROPORTIONS

API Reference

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

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



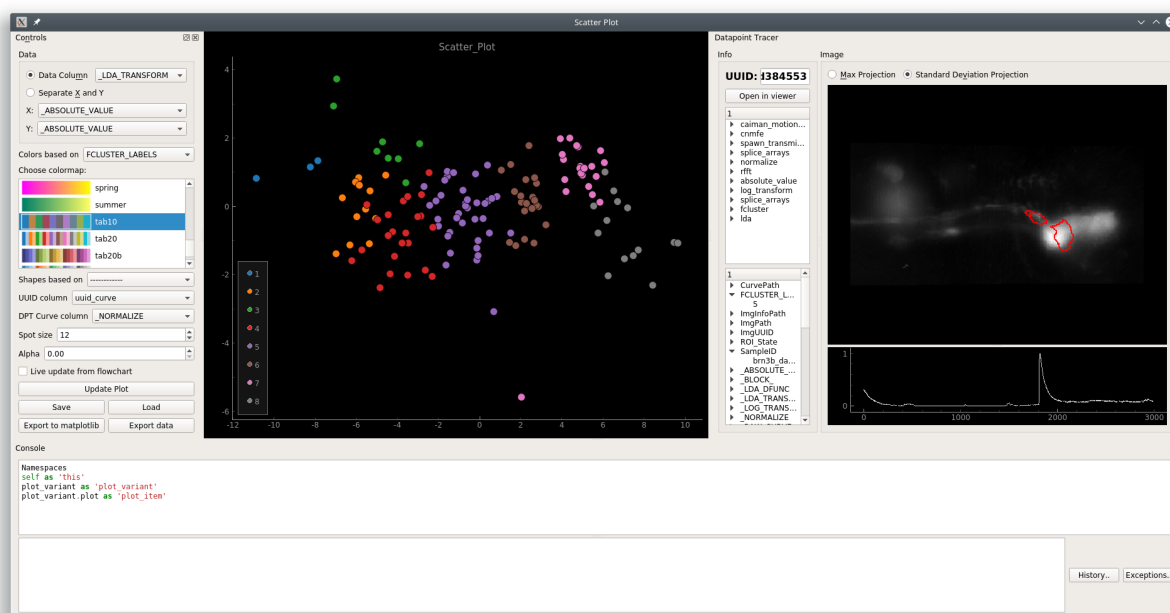
Parameter	Description
X column	DataFrame column containing the categorical labels used for grouping the data Data in each X column sums to 100% if <i>Show percentages</i> is checked
Y column	DataFrame column containing the categorical labels that are counted for each group
Show percentages	When unchecked 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

API Reference

Interactive scatter plot

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

31.1 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

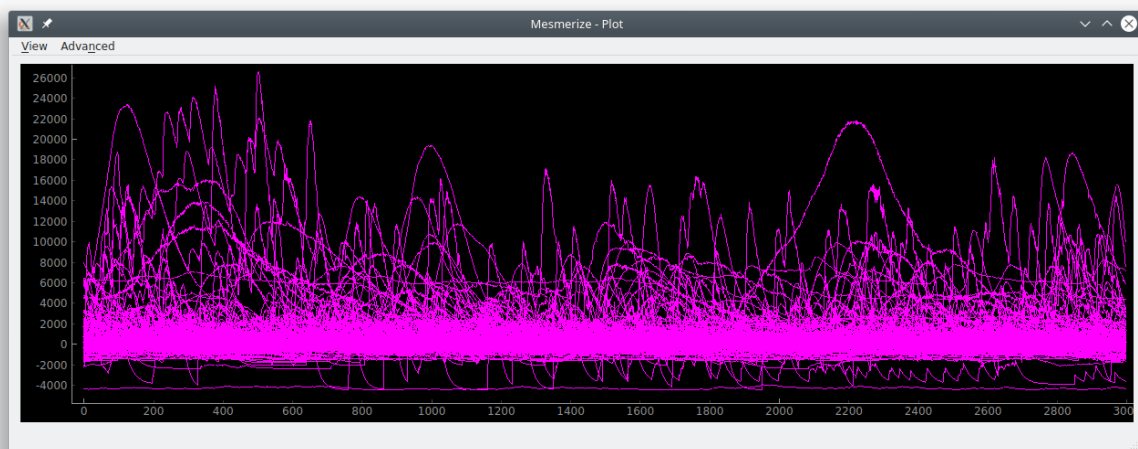
Below the plot: Status label that displays plotting issues. Click the label to see more information.

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.

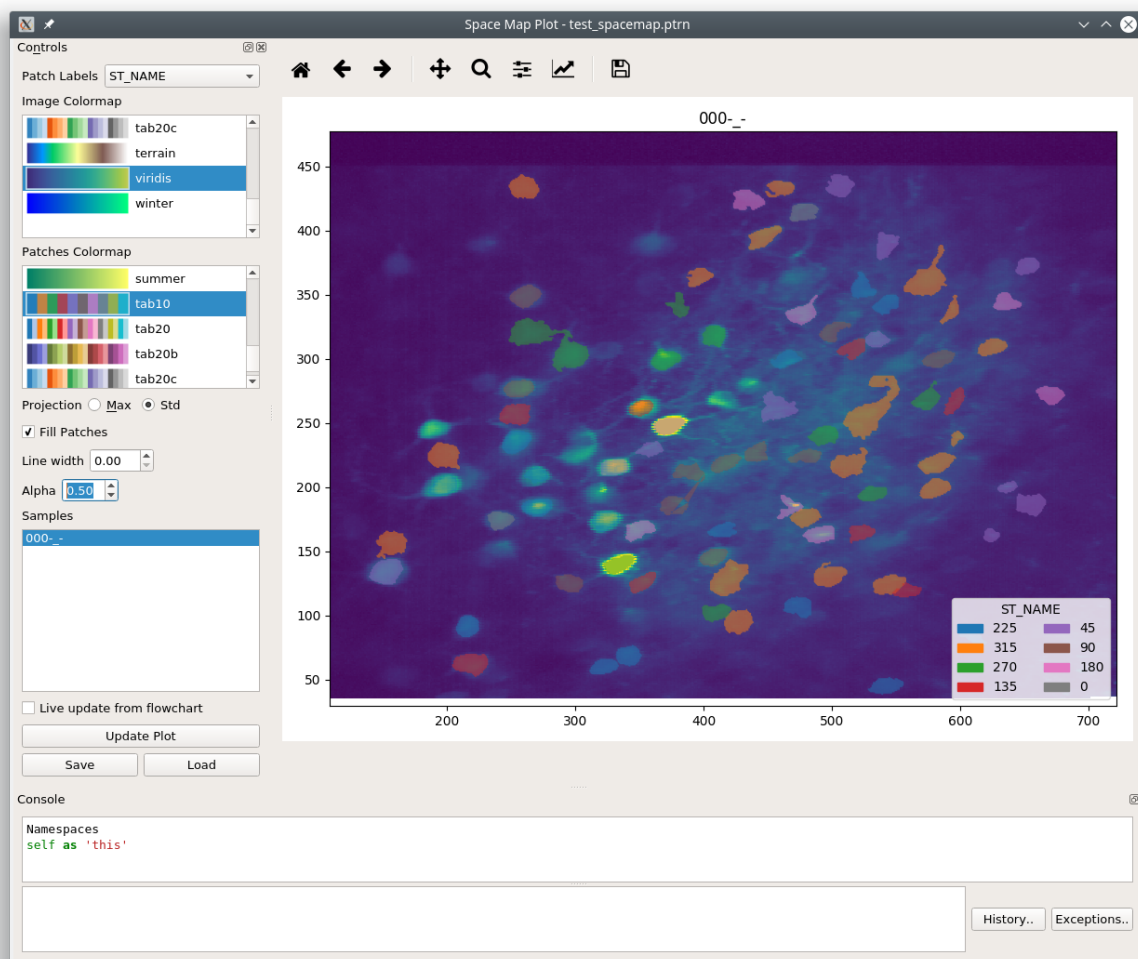


SPACEMAP

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>

33.1 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.

33.2 Console

See also:

API Reference

33.2.1 Namespace

reference	Description
this	The <i>SpaceMapWidget</i> 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>

33.2.2 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()
```

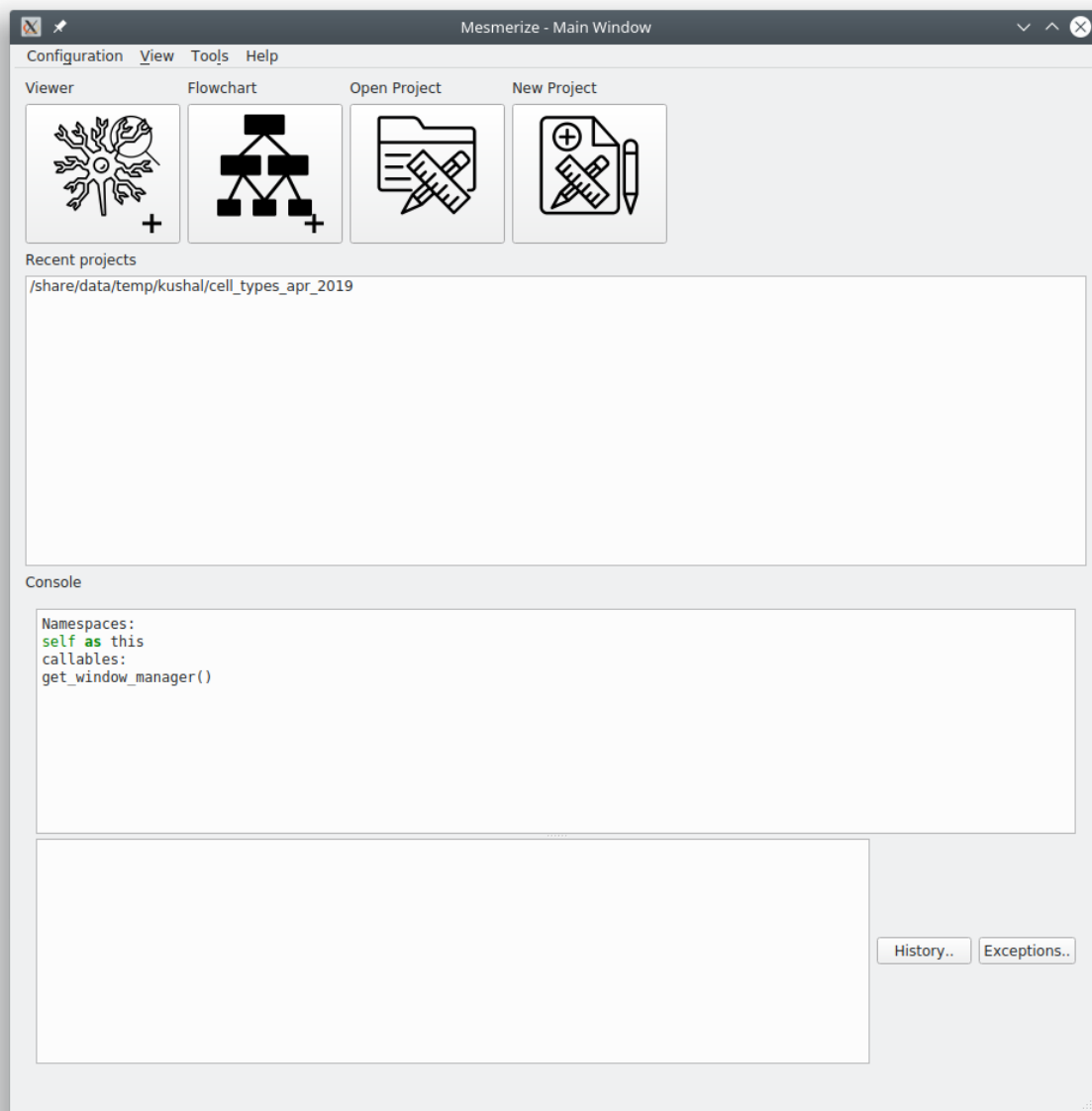

WELCOME WINDOW

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
dataframes	Contains an file storing the project dataframe, root.dfr, and backups. A new backup is created every time a a new <i>Sample</i> is <i>added to the project</i> . 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 <i>Batch Manager</i> if you wish.
flowcharts	Used for storing .fc flowchart files that save the layout of nodes in a flowchart.
plots	Used for storing .ptrn interactive plot files.

See also:

Flowchart Overview and *Saving Plots*

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

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.

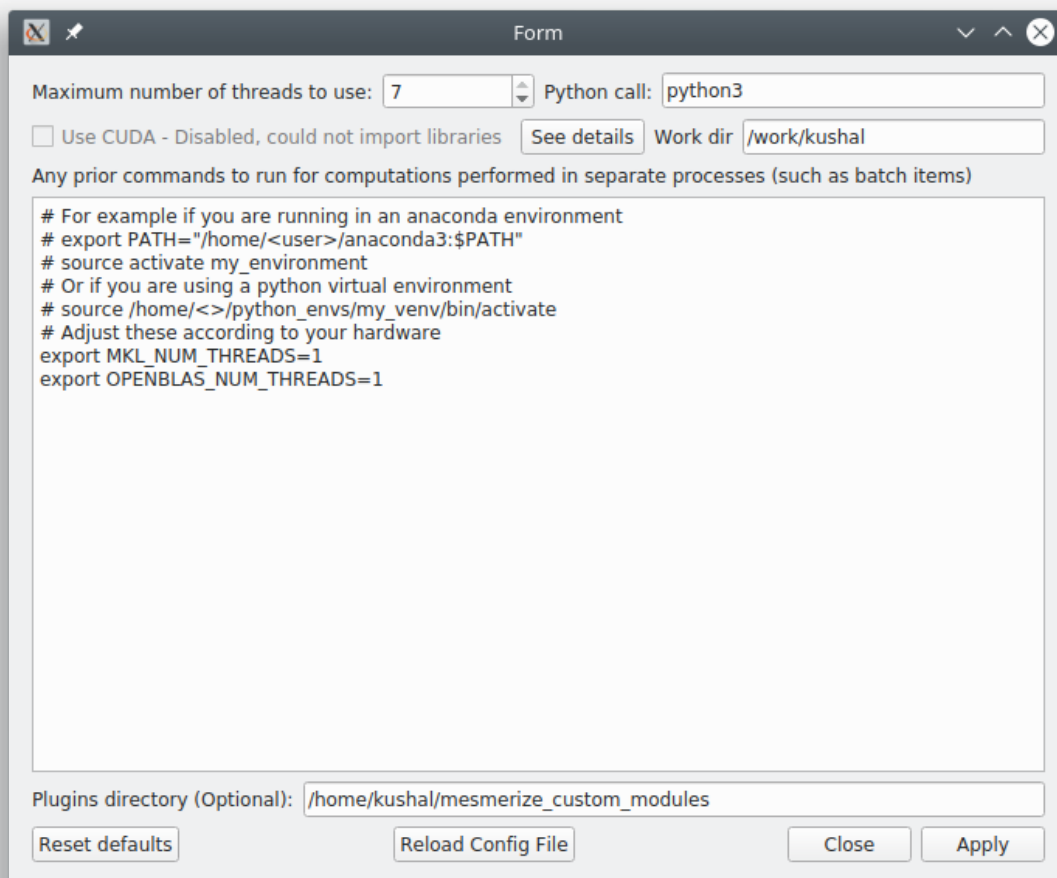
Edit axis, curve...: For some plots. Options for formatting x & y axis limits, labels, select line style, color, etc.

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".
- 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 proccesses 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. `python3` should work for Linux & Mac OSX. We've found that this needs to be set to `python` to work within Anaconda environments on Windows.

Use CUDA: Use CUDA acceleration if you have a GPU with CUDA cores. You must have `pycuda` and `scikit-cuda` (as well as the [nvidia CUDA toolkit](#)) installed. CUDA acceleration isn't used much currently.

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 or conda environment you will need activate the environment so you must include the line `source /path_to_venv/bin/activate` or `conda activate <env_name>` to the pre-run commands.
- 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.

NODES

The easiest way to create a new node is create a class that inherits from *CtrlNode*. You can place this class in one of the existing modules in `mesmerize/pyqtgraphCore/flowchart/library`

Become familiar with the *Transmission object* before creating a node. Almost all nodes work with a *Transmission* object for storing data. Make sure to conform to the conventions for naming of data columns and categorical columns.

40.1 Simple node

The simplest type of node performs an operation on a user-specified data column and doesn't take any parameters.

Basic structure

```
class MyNode(CtrlNode):
    """Doc String that is shown when node is clicked on"""
    nodeName = 'MyNode'
    uiTemplate = <list of tuples, see below>

    def processData(self, transmission: Transmission):
        self.t = transmission.copy()  #: input to this node

        # .. do stuff to the Transmission DataFrame

        params = <dict of analysis params>

        # log the analysis that was done
        self.t.history_trace.add_operation('all', 'mynode', params)

        # Send the output
        return self.t
```

Required class attributes:

- **nodeName: (str)** The name prefix used when instances of this node are created in the flowchart
- **uiTemplate: (list)** *List of UI element tuples (see section)*

If the node only has one input and one output terminal it is sufficient to create a `processData` method that performs the node's analysis operation(s).

40.1.1 Example

```

1 class Derivative(CtrlNode):
2     """Return the Derivative of a curve."""
3     nodeName = 'Derivative'
4     uiTemplate = [('data_column', 'combo', {}),
5                   ('Apply', 'check', {'checked': False, 'applyBox': True})
6                   ]
7
8     # If there is only one input and one output terminal, processData will
9     # always have a single argument which is just the input transmission,
10    # i.e. the output from the previous node.
11    def processData(self, transmission: Transmission):
12        # the input transmission
13        self.t = transmission
14
15        # If a comboBox widget named 'data_column' is specified in the
16        # uiTemplate, you can update its contents using the following method.
17        # This will populate the comboBox with all the data columns from the
18        # input transmission and select the input data column as the
19        # output data column from the previous node.
20        self.set_data_column_combo_box()
21
22        # Check if the Apply checkbox is checked
23        if self.ctrls['Apply'].isChecked() is False:
24            return
25
26        # Make a copy of the input transmission so we can modify it to create an output
27        self.t = transmission.copy()
28
29        # By convention output columns are named after the node's name and in all caps
30        # Columns containing numerical data have a leading underscore
31        output_column = '_DERIVATIVE'
32
33        # Perform this node's operation
34        self.t.df[output_column] = self.t.df[self.data_column].apply(np.gradient)
35
36        # Set transmission's `last_output` attribute as the name of the output column
37        # This is used by the next node to know what the last output data was
38        self.t.last_output = output_column
39
40        # Create a dict of parameters that this node used
41        # Usually a dict that captures the state of the uiTemplate
42        # the transmission `last_unit` attribute is the data units of the data
43        # in the output column (i.e. `t.last_output`). Change it only if the data units
44        ↪ change
45        params = {'data_column': self.data_column,
46                  'units': self.t.last_unit
47                  }
48
49        # Add a log of this node's operation to the transmission's `HistoryTrace` instance
50        # Nodes usually perform an operation on all datablocks pass 'all' to the data_
51        ↪ block_id argument
52        # By convention the operation name is the name of the node in lowercase letters
53        self.t.history_trace.add_operation(data_block_id='all', operation='derivative',
54        ↪ parameters=params)

```

(continues on next page)

(continued from previous page)

```

53     # return the modified transmission instance, which is then the output of this node
54     return self.t

```

40.2 Complex node

For a more complex node with multiple inputs and/or outputs you will need to explicitly specify the terminals when instantiating the parent *CtrlNode* and create a simple override of the `process()` method.

Format of the dict specifying the node's terminals:

```

{
    <terminal name (str)>:          {'io': <'in' or 'out'>},
    <another terminal name (str)>:  {'io', <'in' or 'out'>},
    <another terminal name (str)>:  {'io', <'in' or 'out'>}
    ...
}

```

Override the `process()` method simply pass all kwargs to a `processData()` method and return the output. The `processData()` method must return a dict. This dict must have keys that correspond to the specified output terminals. The values of these keys are the outputs from the respective terminals.

Here is a trimmed down example from the *LDA* node:

```

1  class LDA(CtrlNode):
2      """Linear Discriminant Analysis, uses sklearn"""
3      nodeName = "LDA"
4      uiTemplate = [('train_data', 'list_widget', {'selection_mode': QtWidgets.
5          ↳QAbstractItemView.ExtendedSelection,
6              'toolTip': 'Column containing the_
7          ↳training data'})],
8          ('train_labels', 'combo', {'toolTip': 'Column containing training labels
9          ↳'}),
10         ('solver', 'combo', {'items': ['svd', 'lsqr', 'eigen']}),
11         ('shrinkage', 'combo', {'items': ['None', 'auto', 'value']}),
12         ('shrinkage_val', 'doubleSpin', {'min': 0.0, 'max': 1.0, 'step': 0.1,
13          ↳'value': 0.5}),
14         ('n_components', 'intSpin', {'min': 2, 'max': 1000, 'step': 1, 'value':_
15          ↳2}),
16         ('tol', 'intSpin', {'min': -50, 'max': 0, 'step': 1, 'value': -4}),
17         ('score', 'lineEdit', {}),
18         ('predict_on', 'list_widget', {'selection_mode': QtWidgets.
19          ↳QAbstractItemView.ExtendedSelection,
20              'toolTip': 'Data column of the input
21          ↳"predict" Transmission\n'
22              'that is used for predicting_
23          ↳from the model'}),
24         ('Apply', 'check', {'applyBox': True, 'checked': False})
25     ]
26
27     def __init__(self, name, **kwargs):
28         # Specify the terminals with a dict
29         CtrlNode.__init__(self, name, terminals={'train': {'io': 'in'},
30             'predict': {'io': 'in'},

```

(continues on next page)

(continued from previous page)

```

24         'T': {'io': 'out'},
25         'coef': {'io': 'out'},
26         'means': {'io': 'out'},
27         'predicted': {'io': 'out'}
28     },
29     **kwargs)
30     self.ctrls['score'].setReadOnly(True)
31
32     # Very simple override
33     def process(self, **kwargs):
34         return self.processData(**kwargs)
35
36     def processData(self, train: Transmission, predict: Transmission):
37         self.t = train.copy()  #: Transmisison instance containing the training data with
38         ↪the labels
39         self.to_predict = predict.copy()  #: Transmission instance containing the data to
40         ↪predict after fitting on the the training data
41
42         # function from mesmerize.analysis.utils
43         dcols, ccols, ucols = organize_dataframe_columns(self.t.df.columns)
44
45         # Set available options for training data & labels
46         self.ctrls['train_data'].setItems(dcols)
47         self.ctrls['train_labels'].setItems(ccols)
48
49         dcols = organize_dataframe_columns(self.to_predict.df.columns)
50         # Set available data column options for predicting on
51         self.ctrls['predict_on'].setItems(dcols)
52
53         # Process further only if Apply is checked
54         if not self.ctrls['Apply'].isChecked():
55             return
56
57         # Get the user-set parameters
58         train_column = self.ctrls['train_data'].currentText()
59
60         # ... get other params
61         n_components = self.ctrls['n_components'].value()
62
63         # ... do stuff
64
65         # This node ouputs separate transmissions that are all logged
66         self.t.history_trace.add_operation('all', 'lda', params)
67         self.t_coef.history_trace.add_operation('all', 'lda', params)
68         self.t_means.history_trace.add_operation('all', 'lda', params)
69
70         # the `to_predict` transmission is logged differently
71         self.to_predict.history_trace.add_operations('all', 'lda-predict', params_predict)
72
73         # dict for organizing this node's outputs
74         # The keys MUST be the same those specified for this node's output terminals
75         out = {'T': self.t,
76               'coef': self.t_coef,
77               'means': self.t_means,
78               'predicated': self.to_predict
79             }

```

(continues on next page)

(continued from previous page)

79

`return out`

40.3 uiTemplate

Specify the uiTemplate attribute as a list of tuples.

One tuple per UI element with the following structure:

(<name (str)>, <type (str)>, <dict of attributes to set>)

Examples:

```
('dist_metric', 'combo', {'items': ['euclidean', 'wasserstein', 'bah'], 'toolTip':  
↪ 'distance metric to use'})  
( 'n_components', 'intSpin', {'min': 2, 'max': 10, 'value': 2, 'step': 1, 'toolTip':  
↪ 'number of components'})  
( 'data_columns', 'list_widget', {'selection_mode': QtWidgets.QAbstractItemView.  
↪ ExtendedSelection})
```

The name can be anything. Accepted types and accepted attributes are outlined below

widget type	attributes that can be set
intSpin	<i>min (int)</i> : minimum value allowed in the spinbox <i>max (int)</i> : maximum value allowed <i>step (int)</i> : step size <i>value (int)</i> : default value
doubleSpin	<i>min (float)</i> : minimum value allowed in the spinbox <i>max (float)</i> : maximum value allowed <i>step (float)</i> : step size <i>value (float)</i> : default value
check	<i>checked (bool)</i> : default state of the checkBox <i>applyBox (bool)</i> : Whether this is an “Apply checkbox”
radioBtn	<i>checked (bool)</i> : default state of this radioButton
combo	<i>items (list)</i> : default list of items that will be set in the comboBox
list_widget	<i>items (list)</i> : default list of items that will be set in the list_widget <i>selection_mode</i> : One of the accepted QAbstractItemView selection modes
lineEdit	<i>text (str)</i> : default text in the line edit <i>placeholder (str)</i> : placeholder text <i>readOnly (bool)</i> : set as read only
plainTextEdit	<i>text (str)</i> : default text in the text edit <i>placeholder (str)</i> : placeholder text
label	<i>text (str)</i> : default text
button	<i>text (str)</i> : default text on the button <i>checkable (bool)</i> : whether this button is checkable
color	Does not take any attributes

All UI widget types outlined above take ‘toolTip’ as an attribute which can be used to display tooltips

PLOTS

The easiest way to create a plot module is by subclassing the *BasePlotWidget*. You could also subclass the abstract base if you need to define all the common functionality differently.

41.1 General Design

This shows how you can design a plot using the *SpaceMapPlot* as a simple example. It will generally consist of a class for the main plot area, plot control, and the plot window which contains the controls and plot area.

41.2 Plot Area

A class which holds the actual plot, could be a matplotlib widget or pyqtgraph plot widget for example. In the *SpaceMapPlot* this is simply a subclass of the pyqtgraph matplotlib widget with a few more attributes and a helper method. The *error_label* attribute is simply a QLabel used for displaying a plot error summary and is handled by the *exceptions_label* decorator from *qdialogs*.

41.3 Plot Controls

A class which manages the plot controls. Generally useful to use a QDockWidget for this and design the actual GUI layout using QtDesigner. The *WidgetRegistry* provides a simple way to package the plot control values (plot parameters) into a dict.

Register a widget to the registry using the *WidgetRegistry* instance's *register()* method. The **getter** method corresponds to the widget's method which will return the value of the widget (such as text or a number) that is set in the parameters dict which is created when *widget_registry.get_state()* is called. Correspondingly, **setter** method is the widget's method that is used to set a value to the widget and is used when saved plots are restored. In essence, **setter** and **getter** must be interoperable.

The Space Map plot uses a *sig_changed* class attribute that simply emits when any of the widgets are changed. This is later used in the main plot window to update the plot.

A *fill_widget()* method is useful for populating the controls in the dock widget when the input data to the plot window changes.

In the Space Map widget, *get_state()* and *set_state()* simply wrap the corresponding methods from the *WidgetRegistry* instance.

41.4 Plot Window

Subclass from `QMainWindow` and *BasePlotWidget*. **Mandatory** to specify a *drop_opts* class attribute of type *list*. This list contains the name of any widgets in the dict return from the *WidgetRegistry* that should be excluded when saving the plot. This should be used if you are using data types that are not JSON serializable, however it is rarely necessary. Support for *drop_opts* may be removed in the future.

In general specifying the methods described below should be sufficient to create a saveable plot. If you need finer control of the data struture for saving/opening plots you can subclass from the *abstract base class*.

41.4.1 `__init__`

Setting things up, connection signals, etc. Useful to have a console dock widget.

41.4.2 `set_update_live()`

A method that interacts with a “live update” checkbox in the plot controls.

41.4.3 `set_input()`

Set the input transmission for this plot if it is in “live update” mode or if the plot instance is new (has not had input data previously).

Useful to have a *BasePlotWidget.signal_blocker* decorator so that the plot doesn’t constantly update while the new data comes in, since it could cause plot options to change etc.

41.4.4 `fill_control_widget()`

Organize the plot options that are available to the user and set the control widgets.

Useful to have a *BasePlotWidget.signal_blocker* decorator here as well for same reasons as described above.

41.4.5 `update_plot()`

This is the core of plot. Use the input transmission and the user-selected plot parameters to draw the plot in the plot area. Generally interacts with the Plot Area instance. You can use the *get_state()* method of the control widget’s *WidgetRegistry* to conveniently get a dict of all the user-selected plot parameters.

Useful to have an *exceptions_label* or *present_exceptions* decorator from the *qdialogs module*. The *exceptions_label* provides a less annoying way to present exceptions that occurred when updating the plot.

41.4.6 `get_plot_opts()`

Usually just returns the dict from the widget registry containing all user-set plot parameters.

41.4.7 `set_plot_opts()`

Usually just calls the widget registry's `set_state()` method to set the plot parameters from a dict.

Useful to have a *BasePlotWidget*.*signal_blocker* decorator. In general you would use the *BasePlotWidget*.*open_plot()* method to open a saved plot and it takes care of updating the plot after the input transmission and plot parameters are set.

41.4.8 `show_exception_info()`

Called when the *exceptions_label* is clicked. Opens a QMessageBox to show the entire stack trace.

VIEWER MODULES

Viewer modules appear as either QDockWidgets or QWidgets to the user. They must consist of a main `ModuleGUI` class which inherits from either `QDockWidget` or `QWidget`. They can utilize any additional python modules, classes, etc.

42.1 Instructions

1. Create a directory plugins directory if you don't have one. If you are using a snap installation this has to be within your home folder. Set this plugins directory in the *System Configuration*. This directory can contain as many custom modules as you want. All python modules within the plugins directory are automatically imported.
2. Download the `__init__.py` and place it within the plugins directory.
3. Create the main module file for your custom module. This file can be named as you wish and must use the struture outlined below. In addition to this main module file you can create a directory to house any other modules, files etc. You can create Qt templates using Qt Creator and convert them to .py template files using `pyuic5` and use them for your custom module.

Basic Structure

```
1  from PyQt5 import QtWidgets
2
3  module_name = 'Example Module'
4
5  # You must define module_name.
6  # This is the name that will be displayed in the "Plugins" menu of the
   ↳ Viewer Window.
7  # You can use this to reference the ModuleGUI instance through the Viewer
   ↳ Console via ``get_module(<module_name>``
8
9
10 # The main GUI class MUST be named ModuleGUI.
11 # You can have other classes and more GUIs however ModuleGUI is the one
   ↳ that the Viewer Window directly calls.
12
13 class ModuleGUI(QtWidgets.QDockWidget):
14     # The Viewer MainWindow will pass its Viewer instance that can be
   ↳ used to interact with the viewer and work environment.
15     def __init__(self, parent, viewer_instance):
16         QtWidgets.QDockWidget.__init__(self, parent)
17         self.setFloating(True) # Must be floating
```

4. The module will be accessible through the Viewer Window's "Plugins" menu. The names in the plugins menu will correspond to the aforementioned `module_name` variable.

COMMON

`mesmerize.common.get_proj_config(proj_path: str = None) → configparser.RawConfigParser`

Parameters `proj_path` – Full project path

`mesmerize.common.get_project_manager()`
Get the project manager for this Mesmerize instance

`mesmerize.common.get_sys_config()` → dict
Get the user-set system configuration

`mesmerize.common.get_window_manager()`
Get the Window Manager for this Mesmerize instance

43.1 Utils

Some frequently used utility functions

`mesmerize.common.utils.make_workdir(prefix: str = "") → str`

Make a workdir within the `mesmerize_tmp` directory of the workdir specified in the configuration. The name of the created workdir is the date & time of its creation. You can add a prefix to this name.

Parameters `prefix` – Prefix for the workdir name

Returns full workdir path

Return type `str`

`mesmerize.common.utils.make_runfile(module_path: str, savedir: str, args_str: Optional[str] = None, filename: Optional[str] = None, pre_run: Optional[str] = None, post_run: Optional[str] = None) → str`

Make an executable bash script. Used for running python scripts in external processes.

Parameters

- `module_path` (`str`) – absolute module path
- `args_str` (`str`) – str of args that is directly passed with the python command in the bash script
- `savedir` (`Optional[str]`) – working directory
- `filename` (`Optional[str]`) – optional, specific filename for the script
- `pre_run` (`Optional[str]`) – optional, str to run before module is ran
- `post_run` (`Optional[str]`) – optional, str to run after module has run

Returns path to the shell script that can be run

Return type `str`

class `mesmerize.common.utils.HdfTools`

Functions for saving and loading HDF5 data

static `save_dataframe` (*path*: `str`, *dataframe*: `pandas.core.frame.DataFrame`, *metadata*: *Optional*[`dict`] = `None`, *metadata_method*: `str` = `'json'`, *raise_meta_fail*: `bool` = `True`)

Save DataFrame to hdf5 file along with a meta data dict.

Meta data dict can either be serialized with json and stored as a str in the hdf5 file, or recursively saved into hdf5 groups if the dict contains types that hdf5 can deal with. Experiment with both methods and see what works best

Currently the hdf5 method can work with these types: [str, bytes, int, float, np.int, np.int8, np.int16, np.int32, np.int64, np.float, np.float16, np.float32, np.float64, np.float128, np.complex].

If it encounters an object that is not of these types it will store whatever that object's `__str__()` method returns if `on_meta_fail` is False, else it will raise an exception.

Parameters

- **path** (`str`) – path to save the file to
- **dataframe** (`pd.DataFrame`) – DataFrame to save in the hdf5 file
- **metadata** (*Optional*[`dict`]) – Any associated meta data to store along with the DataFrame in the hdf5 file
- **metadata_method** (`str`) – method for storing the metadata dict, either 'json' or 'recursive'
- **raise_meta_fail** (`bool`) – raise an exception if recursive metadata saving encounters an unsupported object If false, it will save the unsupported object's `__str__()` return value

static `load_dataframe` (*filepath*: `str`) → `Tuple`[`pandas.core.frame.DataFrame`, *Optional*[`dict`]]

Load a DataFrame along with meta data that were saved using `HdfTools.save_dataframe`

Parameters **filepath** (`str`) – file path to the hdf5 file

Returns tuple, (DataFrame, meta data dict if present else None)

Return type `Tuple`[`pd.DataFrame`, `Union`[`dict`, `None`]]

static `save_dict` (*d*: `dict`, *filename*: `str`, *group*: `str`, *raise_type_fail*=`True`)

Recursively save a dict to an hdf5 group.

Parameters

- **d** (`dict`) – dict to save
- **filename** (`str`) – filename
- **group** (`str`) – group name to save the dict to
- **raise_type_fail** (`bool`) – whether to raise if saving a piece of data fails

static `load_dict` (*filename*: `str`, *group*: `str`) → `dict`

Recursively load a dict from an hdf5 group.

Parameters

- **filename** (`str`) – filename

- **group** (*str*) – group name of the dict

Returns dict recursively loaded from the hdf5 group

Return type dict

`mesmerize.common.utils.draw_graph` (*l*: List[dict], *filename*: Optional[str] = None, *view*: bool = False) → str

Draw a graph from a list of dicts.

Parameters

- **l** (List[dict]) – list of dicts
- **filename** (Optional[str]) – full path for storing the draw graph pdf file
- **view** (Optional[bool]) – view the graph in the system’s default pdf reader after it is rendered

Returns full path to the graph pdf file

Return type str

43.2 QDialogs

Decorators for Qt Dialog GUIs used throughout Mesmerize

`mesmerize.common.qdialogs.present_exceptions` (*title*: str = 'error', *msg*: str = 'The following error occurred.', *help_func*: callable = None)

Use to catch exceptions and present them to the user in a QMessageBox warning dialog. The traceback from the exception is also shown.

Parameters

- **title** – Title of the dialog box
- **msg** – Message to display above the traceback in the dialog box
- **help_func** – A helper function which is called if the user clicked the “Help” button

`mesmerize.common.qdialogs.exceptions_label` (*label*: str, *exception_holder*: str = None, *title*: str = 'error', *msg*: str = 'The following error occurred')

Use a label to display an exception instead of a QMessageBox

Parameters

- **label** – name of a QLabel instance
- **exception_holder** – name of an exception_holder attribute where the exception message is stored. This can be used to view the whole exception when the label is clicked on for example.
- **title** – title supplied for the QMessageBox (if used later)
- **msg** – message supplied for the QMessageBox (if used later)

`mesmerize.common.qdialogs.use_open_file_dialog` (*title*: str = 'Choose file', *start_dir*: Optional[str] = None, *exts*: List[str] = None)

Use to pass a file path, for opening, into the decorated function using QFileDialog.getOpenFileName

Parameters

- **title** – Title of the dialog box
- **start_dir** – Directory that is first shown in the dialog box.
- **exts** – List of file extensions to set the filter in the dialog box

`mesmerize.common.qdialogs.use_save_file_dialog` (*title: str = 'Save file', start_dir: Optional[str] = None, ext: str = None*)

Use to pass a file path, for saving, into the decorated function using `QFileDialog.getSaveFileName`

Parameters

- **title** – Title of the dialog box
- **start_dir** – Directory that is first shown in the dialog box.
- **exts** – List of file extensions to set the filter in the dialog box

`mesmerize.common.qdialogs.use_open_dir_dialog` (*title: str = 'Open directory', start_dir: Optional[str] = None*)

Use to pass a dir path, to open, into the decorated function using `QFileDialog.getExistingDirectory`

Parameters

- **title** – Title of the dialog box
- **start_dir** – Directory that is first shown in the dialog box.

Example:

```
@use_open_dir_dialog('Select Project Directory', '')
def load_data(self, path, *args, **kwargs):
    my_func_to_do_stuff_and_load_data(path)
```


VIEWER CORE

44.1 ViewerWorkEnv

This objects stores the data that the *Viewer* interacts with.

```
class mesmerize.viewer.core.ViewerWorkEnv (imgdata=None, sample_id="", UUID=None,
                                           meta=None, stim_maps=None,
                                           roi_manager=None, roi_states=None, comments="", origin_file="", custom_cols=None,
                                           history_trace: list = None, additional_data: dict = None, misc: dict = None, **kwargs)
```

`__UUID`

UUID, if opened from a project Sample refers to the ImgUUID

```
__init__ (imgdata=None, sample_id="", UUID=None, meta=None, stim_maps=None,
          roi_manager=None, roi_states=None, comments="", origin_file="", custom_cols=None,
          history_trace: list = None, additional_data: dict = None, misc: dict = None, **kwargs)
```

A class that encapsulates the main work environment objects (img sequence, ROIs, and ROI associated curves) of the viewer. Allows for a work environment to be easily spawned from different types of sources and allows for a work environment to be easily saved in different ways regardless of the type of original data source.

Parameters

- **roi_states** (*dict*) – roi states from ROI Manager module
- **stim_maps** (*dict*) – {‘units’: str, ‘dataframe’: pd.DataFrame}
- **history_trace** (*list*) – list of dicts containing a traceable history of what what done with the work environment, such as params used from modules to process the data

`__weakref__`

list of weak references to the object (if defined)

```
static _organize_meta (meta: dict, origin: str) → dict
```

Organize input meta data dict into a uniform structure :param meta: meta data dict, origin from a json file for example :param origin: name of the origin source of the meta data, such a program or microscope etc. :return: dict organized with keys that are used throughout Mesmerize.

```
clear ()
```

Cleanup of the work environment

```
classmethod from_mesfile (mesfile_object: mesmerize.viewer.core.mesfile.MES,
                          img_reference: str)
```

Return instance of work environment with MesmerizeCore.ImgData class object using seq returned from

MES.load_img from MesmerizeCore.FileInput module and any stimulus map that the user may have specified.

Parameters

- **mesfile_object** – MES object, created from .mes file
- **img_reference** – image reference to load, see `mesmerize.viewer.core.mesfile.MES.get_image_references()`

classmethod from_pickle (*pickle_file_path: str, tiff_path: str = None*)

Get pickled image data from a pickle file & image sequence from a npz or tiff. Used after motion correction & to view a sample from a project DataFrame. Create ImgData class object (See MesmerizeCore.DataTypes) and return instance of the work environment.

Param pickle_file_path: full path to the pickle containing image metadata, stim maps, and roi_states

Param tiff_path: str of the full path to a tiff file containing the image sequence

classmethod from_tiff (*path: str, method: str, meta_path: Optional[str] = None, axes_order: Optional[str] = None, meta_format: Optional[str] = None*)

Return instance of work environment with ImgData.seq set from the tiff file.

Parameters

- **path** – path to the tiff file
- **method** – one of ‘imread’, ‘asarray’, or ‘asarray-multi’. Refers to usage of either tiff-file.imread or tiff-file.asarray. ‘asarray-multi’ will load multi-page tiff files.
- **meta_path** – path to a file containing meta data
- **meta_format** – meta data format, must correspond to the name of a function in viewer.core.organize_meta
- **axes_order** – Axes order as a 3 or 4 letter string for 2D or 3D data respectively. Axes order is assumed to be “txy” or “tzxy” if not specified.

history_trace

history log

imgdata

ImgData instance

isEmpty

Return True if the work environment is empty

static load_mesfile (*path: str*) → `mesmerize.viewer.core.mesfile.MES`

Just passes the path of a .mes file to the constructor of class MES in MesmerizeCore.FileInput. Loads .mes file & constructs MES obj from which individual images & their respective metadata can be loaded to construct viewer work environments using the classmethod viewerWorkEnv.from_mesfile.

Parameters path – full path to a single .mes file.

roi_manager

reference to the back-end ROI Manager that is currently in use

sample_id

SampleID, if opened from a project Sample

stim_maps

Stimulus maps

to_pandas (*proj_path: str, modify_options: Optional[dict] = None*) → *list*

Used for saving the work environment as a project Sample.

Parameters

- **proj_path** – Root path of the current project
- **modify_options** –

Returns list of dicts that each correspond to a single curve that can be appended as rows to the project dataframe

to_pickle (*dir_path: str, filename: Optional[str] = None, save_img_seq=True, UUID=None*) → *str*

Package the current work Env ImgData class object (See MesmerizeCore.DataTypes) and any paramteres such as for motion correction and package them into a pickle & image seq array. Used for batch motion correction and for saving current sample to the project. Image sequence is saved as a tiff and other information about the image is saved in a pickle.

44.2 ImgData

class mesmerize.viewer.core.data_types.**ImgData** (*seq: numpy.ndarray = None, meta: dict = None*)

Object that stores the image sequence and meta data from the imaging source

__init__ (*seq: numpy.ndarray = None, meta: dict = None*)

Parameters

- **seq** – Image sequence as a numpy array, shape is [x, y, t] or [x, y, t, z]
- **meta** – Meta data dict from the imaging source.

44.3 ViewerUtils

The *Viewer* is usually not interacted with directly from modules outside of the viewer (such as viewer modules. They instead use the ViewerUtils class which includes helper functions and a reference to the viewer.

class mesmerize.viewer.core.**ViewerUtils** (*viewer_reference: <module 'mesmerize.pyqtgraphCore.imageview.ImageView' from 'home/docs/checkouts/readthedocs.org/user_builds/mesmerize/envs/v0.2.2/packages/mesmerize-0.2.2-py3.7.egg/mesmerize/pyqtgraphCore/imageview/ImageView.py'>*)

__init__ (*viewer_reference: <module 'mesmerize.pyqtgraphCore.imageview.ImageView' from 'home/docs/checkouts/readthedocs.org/user_builds/mesmerize/envs/v0.2.2/lib/python3.7/site-packages/mesmerize-0.2.2-py3.7.egg/mesmerize/pyqtgraphCore/imageview/ImageView.py'>*)
Some utility functions for interfacing viewer.core.ViewerWorkEnv with the pyqtgraphCore.ImageView widget

__weakref__
list of weak references to the object (if defined)

_clear_workEnv (*clear_sample_id=False*)
Cleanup of the ViewerWorkEnv and ImageView widget

discard_workEnv (*clear_sample_id=False*)

Ask the user if they want to discard their work environment. If Yes, calls `_clear_workEnv()`

set_statusbar (*msg*)

Set the status bar message in the viewer window.

Parameters *msg* – text to display in the status bar

update_workEnv ()

Update the ImageView widget with the ViewerWorkEnv

viewer

reference to the pyqtgraph ImageView widget instance (viewer)

work_env

ViewerWorkEnv instance

44.4 Mesfile

class `mesmerize.viewer.core.mesfile.MES` (*filename: str*)

Handles of opening .mes files and organizing the images and meta data. The `load_img()` method returns a 3D array (dims are [time, cols, rows]) of the image sequence and its associated meta data.

Usage: Create a MES instance by passing the path of your mes file, example:

```
mesfile = MES('/path/to/mesfile/experiment_Feb_31.mes')
```

Call the `get_image_references()` method to get a list of references for images that can be loaded.

To load an image that is available in the instance, just pass one of the references from `get_image_references()` to the `load_img` method:

```
img_array, meta_dict = mesfile.load_img('IF0001_0001')
```

__init__ (*filename: str*)

Parameters *filename* – full path of a single .mes file

__weakref__

list of weak references to the object (if defined)

get_image_references () → *list*

Get a list of all image references available in the instance

load_img (*img_reference: str*) -> (<class 'numpy.ndarray'>, <class 'dict'>)

Parameters *img_reference* – The image reference, usually something like IFxxxx_xxxx or Ifxxxx_xxxx

Returns (image sequence array, meta data dict)

VIEWER MODULES

45.1 Batch Manager

class mesmerize.viewer.modules.batch_manager.**ModuleGUI** (*parent*, *run_batch*: *list* = *None*, *testing*: *bool* = *False*)

GUI for the Batch Manager

__init__ (*parent*, *run_batch*: *list* = *None*, *testing*: *bool* = *False*)

Initialize self. See help(type(self)) for accurate signature.

add_item (*module*: *str*, *input_workEnv*: *mesmerize.viewer.core.viewer_work_environment.ViewerWorkEnv*,
input_params: *dict*, *name*: *str* = "", *info*: *dict* = "") → *uuid.UUID*

Add an item to the currently open batch

Parameters

- **module** – The module to run from /batch_run_modules.
- **input_workEnv** – Input workEnv that the module will use
- **input_params** – Input params that the module will use. Depends on your subclass of BatchRunInterface.process() method
- **name** – A name for the batch item
- **info** – A dictionary with any metadata information to display in the scroll area label.

Returns UUID of the added item

del_item ()

Delete the currently selected item from the batch and any corresponding dependents of the item's output

df

pandas.DataFrame that stores a “database” of information on the batch

get_item_index (*u*: *Union[uuid.UUID, str]*) → *int*

Get DataFrame index from UUID

Parameters *u* – UUID or str representing UUID

Returns numerical index of the DataFrame corresponding to the UUID

load_item_input (*viewers*: *Union[mesmerize.viewer.main_window.MainWindow, collections.UserList]*, *r*: *pandas.core.series.Series* = *None*, *UUID*: *uuid.UUID* = *None*)

Pass either the batch DataFrame row or UUID of the item of which to load the input into a viewer

Parameters

- **viewers** – ViewerWindow or list of ImageView

- **r** – Row of batch DataFrame corresponding to the selected item
- **UUID** – UUID of the item to load input from

load_item_output (*module: str, viewers: Union[mesmerize.viewer.main_window.MainWindow, mesmerize.pyqtgraphCore.imageview.ImageView.ImageView, collections.UserList], UUID: uuid.UUID*)
Calls subclass of BatchRunInterface.show_output()

Parameters

- **module** – The module name under /batch_run_modules that the batch item is from
- **viewers** – ViewerWindow, ImageView, or list of ViewerWindows
- **UUID** – UUID of the item to load output from

process_batch (*start_ix: Union[int, uuid.UUID] = 0, clear_viewers=False*)
Process everything in the batch by calling subclass of BatchRunInterface.process() for all items in batch

Parameters

- **start_ix** – Either DataFrame index (int) or UUID of the item to start from.
- **clear_viewers** – Clear work environments in all viewers that are open

show_item_info (*s: PyQt5.QtWidgets.QListWidgetItem*)
Shows any info (such as the batch module's params) in the meta-info label

45.2 Tiff Module

Uses the **tiff** package created by Christoph Gohlke: [*https://pypi.org/project/tiff/*](https://pypi.org/project/tiff/)

Can be used with scripts within Mesmerize for loading tiff files without using the API of *Viewer Core*

class mesmerize.viewer.modules.tiff_io.**ModuleGUI** (*parent, viewer_reference*)

check_meta_path () → *bool*
check if a file exists with the same name and the meta data extension specified by the selected meta data format

load (*tiff_path: str, method: str, axes_order: Optional[str] = None, meta_path: Optional[str] = None, meta_format: Optional[str] = None*) → *mesmerize.viewer.core.viewer_work_environment.ViewerWorkEnv*
Load a tiff file along with associated meta data

Parameters

- **tiff_path** – path to the tiff file
- **meta_path** – path to the json meta data file
- **method** – one of “asarray”, “asarray-multi”, or “imread” “asarray” and “asarray-multi” uses `tiffio.asarray()` “asarray-multi” is for multi-page tiffs “imread” uses `tiffio.imread()`
- **axes_order** – axes order, examples: txy, xyt, tzxy, xytz etc.
- **meta_format** – name of function from viewer.core.organize_meta that should be used to organize the meta data.

45.3 Caiman Motion Correction

Front-end for [Caiman NoRMCorre](#) parameters entry

Can be used with scripts for adding batch items.

See also:

User guide

```
class mesmerize.viewer.modules.caiman_motion_correction.ModuleGUI (parent,  
                                                                    viewer_reference)
```

```
get_params (group_params: bool = False) → dict
```

Get a dict of the set parameters :return: parameters dict :rtype: dict

```
add_to_batch_elas_corr ()
```

Add a batch item with the currently set parameters and the current work environment.

45.4 CNMF

Front-end for [Caiman CNMF](#) parameter entry

Can be used with scripts for adding batch items.

See also:

User guide

```
class mesmerize.viewer.modules.cnmf.ModuleGUI (parent, viewer_reference)
```

```
get_params (*args, group_params: bool = False) → dict
```

Get a dict of the set parameters. If the work environment was loaded from a motion correction batch item it put the `bord_px` in the dict. Doesn't use any arguments

Returns parameters dict

Return type *dict*

```
add_to_batch (params: dict = None) → uuid.UUID
```

Add a CNMF batch item with the currently set parameters and the current work environment.

45.5 CNMFE

Front-end for [Caiman CNMFE](#) parameter entry

Can be used with scripts for adding batch items.

See also:

User guide

```
class mesmerize.viewer.modules.cnmfe.ModuleGUI (parent, viewer_reference)
```

```
get_params (item_type: str, group_params: bool = False) → dict
```

Get a dict of the set parameters. If the work environment was loaded from a motion correction batch item it put the `bord_px` in the dict. Doesn't use any arguments

Parameters **item_type** – one of *corr_pnr* or *cnmfe*

add_to_batch_corr_pnr (*params: dict = None*) → [uuid.UUID](#)

Add a Corr PNR batch item with the currently set parameters and the current work environment.

add_to_batch_cnmfe (*params: dict = None*) → [uuid.UUID](#)

Add a CNMFE batch item with the currently set parameters and the current work environment.

ROI MANAGER

The GUI QDockWidget that communicates with the *back-end managers*

46.1 ModuleGUI

```
class mesmerize.viewer.modules.roi_manager.ModuleGUI (parent, viewer_reference)
    The GUI front-end for the ROI Manager module

    __init__ (parent, viewer_reference)
        Instantiate attributes

    manager
        The back-end manager instance.

    eventFilter (QObject, QEvent)
        Set some keyboard shortcuts

    slot_delete_roi_menu ()
        Delete the currently selected ROI

    start_backend (type_str: str)
        Choose backend, one of the Manager classes in the managers module.

    start_manual_mode ()
        Start in manual mode. Creates a new back-end manager instance (Uses ManagerManual)

    add_manual_roi (shape: str)
        Add a manual ROI. Just calls ManagerManual.add_roi

    package_for_project () → dict
        Gets all the ROI states so that they can be packaged along with the rest of the work environment to be
        saved as a project Sample

    set_all_from_states (states: dict)
        Set all the ROIs from a states dict. Instantiates the appropriate back-end Manager

    import_from_imagej ()
        Import ROIs from ImageJ zip file
```

46.2 Managers

The back-end managers that are used by the *ROI Manager ModuleGUI*

The managers hold instances of *ROIs* in an instance of *ROIList*

46.2.1 AbstractBaseManager

```
class mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager (parent,
                                                                                     ui,
                                                                                     viewer_interface: mesmerize.viewer.core.common.ViewerUtils)
```

Base ROI Manager

```
__init__ (parent, ui, viewer_interface: mesmerize.viewer.core.common.ViewerUtils)
    Set the common attributes
```

Parameters

- **parent** – The ModuleGUI QDockWidget instance
- **ui** – The ui of the ModuleGUI QDockWidget instance,
- **viewer_interface** – A ViewerUtils instance for accessing the Viewer the parent QDockWidget belongs to

```
roi_list
    The ROIList instance that stores the list of ROIs
```

```
abstract add_roi (*args, **kwargs)
    Method for adding an ROI, must be implemented in subclass
```

```
is_empty () → bool
    Return true if the ROI list is empty, else return False
```

```
get_all_states () → dict
    Get the ROI states for all ROIs in self.roi_list so that they can be restored. The appropriate manager is instantiated based on the 'roi_type' key of the returned dict
```

```
get_plot_item () → mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem
    Get the viewer plot item that is associated to these ROIs
```

```
clear ()
    Cleanup of all ROIs in the list
```

```
__del__ ()
    Cleanup of all ROIs in the list and deletes the manager instance. Used when switching modes.
```

```
__weakref__
    list of weak references to the object (if defined)
```

46.2.2 ManagerManual

```
class mesmerize.viewer.modules.roi_manager_modules.managers.ManagerManual (parent,  
                                                                    ui,  
                                                                    viewer_interface)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.managers.
AbstractBaseManager`

The Manager for the Manual mode

```
__init__ (parent, ui, viewer_interface)  
    Set the common attributes
```

Parameters

- **parent** – The ModuleGUI QDockWidget instance
- **ui** – The ui of the ModuleGUI QDockWidget instance,
- **viewer_interface** – A ViewerUtils instance for accessing the Viewer the parent QDockWidget belongs to

```
create_roi_list ()  
    Create a new empty ROI list instance for storing Manual ROIs
```

```
add_roi (shape: str) → mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI  
    Add an ROI to the list
```

Parameters **shape** – either “PolyLineROI” or “EllipseROI”

```
restore_from_states (states: dict)  
    Restore ROIs from states
```

```
get_all_states () → dict  
    Get the ROI states so that they can be restored later
```

```
import_from_imagej (path: str)  
    Uses read-roi package created by Hadrien Mary. https://pypi.org/project/read-roi/
```

Parameters **path** – Full path to the ImageJ ROIs zip file

46.2.3 ManagerScatterROI

```
class mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScatterROI (parent,  
                                                                    ui,  
                                                                    viewer_interface:  
                                                                    mes-  
                                                                    mer-  
                                                                    ize.viewer.core.com)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.managers.
AbstractBaseManager`

Manager for unmoveable ROIs drawn using scatterplots

```
__init__ (parent, ui, viewer_interface: mesmerize.viewer.core.common.ViewerUtils)  
    Set the common attributes
```

Parameters

- **parent** – The ModuleGUI QDockWidget instance
- **ui** – The ui of the ModuleGUI QDockWidget instance,

- **viewer_interface** – A ViewerUtils instance for accessing the Viewer the parent QDockWidget belongs to

add_roi (*curve: numpy.ndarray, xs: numpy.ndarray, ys: numpy.ndarray, metadata: dict = None, dfof_data: numpy.ndarray = None, spike_data: numpy.ndarray = None*) → *mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI*

Add a single ROI

xs and *ys* arguments are 1D numpy arrays.

Parameters

- **curve** – curve data, 1-D array, y values/intensity values
- **xs** – x-values for the scatter plot to spatially illustrate the ROI
- **ys** – corresponding y-values for the scatter plot to spatially illustrate the ROI
- **metadata** – Any metadata for this ROI

Returns ScatterROI object

restore_from_states (*states: dict*)

Restore from states, such as when these ROIs are saved with a Project Sample

create_roi_list ()

Create empty ROI List

set_spot_size (*size: int*)

Set the spot size for the scatter plot which illustrates the ROI

46.2.4 ManagerVolROI

```
class mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolROI (parent,  
                                                                           ui,  
                                                                           viewer_interface:  
                                                                           mes-  
                                                                           mer-  
                                                                           ize.viewer.core.common.
```

Bases: *mesmerize.viewer.modules.roi_manager_modules.managers.
ManagerScatterROI*

Manager for 3D ROIs

__init__ (*parent, ui, viewer_interface: mesmerize.viewer.core.common.ViewerUtils*)

Set the common attributes

Parameters

- **parent** – The ModuleGUI QDockWidget instance
- **ui** – The ui of the ModuleGUI QDockWidget instance,
- **viewer_interface** – A ViewerUtils instance for accessing the Viewer the parent QDockWidget belongs to

set_zlevel (*z: int*)

Set the current z-level to be visible in the viewer

create_roi_list ()

Create new empty ROI list

46.2.5 ManagerVolCNMF

class `mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMF` (*parent*,
ui,
viewer_interface)

Bases: `mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolROI`

Manager for 3D CNMF based ROIs

__init__ (*parent*, *ui*, *viewer_interface*)

Set the common attributes

Parameters

- **parent** – The ModuleGUI QDockWidget instance
- **ui** – The ui of the ModuleGUI QDockWidget instance,
- **viewer_interface** – A ViewerUtils instance for accessing the Viewer the parent QDockWidget belongs to

create_roi_list ()

Create new empty ROI list

add_all_components (*cnmf_data_dict*: *dict*, *input_params_dict*: *dict*)

Add all components from a CNMF(E) output. Arguments correspond to CNMF(E) outputs

Parameters

- **cnmf_data_dict** – CNMF results data directly from the HDF5 file
- **input_params_dict** – dict of input params, from the batch manager
- **calc_raw_min_max** – Calculate raw min & max for each ROI

Returns

add_roi ()

Not implemented, uses `add_all_components` to import all ROIs instead

restore_from_states (*states*: *dict*)

Restore from states, such as when these ROIs are saved with a Project Sample

get_all_states () → *dict*

Get all states so that they can be restored

update_idx_components (*ix*: *int*)

Update `idx_components` if the user manually delete an ROI

set_spot_size (*size*: *int*)

Set the spot size for the scatter plot which illustrates the ROI

46.2.6 ManagerCNMFROI

class `mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNMFROI` (*parent*,
ui,
viewer_interface)

Bases: `mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager`

Manager for ROIs imported from CNMF or CNMFE outputs

`__init__ (parent, ui, viewer_interface)`

Instantiate necessary attributes

`create_roi_list ()`

Create empty CNMFROI list

`add_all_components (cnmf_data_dict, input_params_dict, calc_raw_min_max=False)`

Add all components from a CNMF(E) output. Arguments correspond to CNMF(E) outputs

Parameters

- **cnmf_data_dict** – CNMF results data directly from the HDF5 file
- **input_params_dict** – dict of input params, from the batch manager
- **calc_raw_min_max** – Calculate raw min & max for each ROI

Returns

`add_roi ()`

Not implemented, uses `add_all_components` to import all ROIs instead

`restore_from_states (states: dict)`

Restore from states, such as when these ROIs are saved with a Project Sample

`get_all_states ()` → dict

Get all states so that they can be restored

`update_idx_components (ix: int)`

Update `idx_components` if the user manually delete an ROI

46.3 ROI List

Used for holding instance of *ROIs*

```
class mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList (ui,
                                                                    roi_types:
                                                                    type,
                                                                    viewer_interface:
                                                                    mesmer-
                                                                    ize.viewer.core.common.ViewerU
```

A list for holding ROIs of one type

`__init__ (ui, roi_types: type, viewer_interface: mesmerize.viewer.core.common.ViewerUtils)`

Instantiate

Parameters

- **ui** – The ui from the parent ModuleGUI, used to interact with the ROI list widget etc.
- **roi_types** – The type of ROI that this list will hold
- **viewer_interface** – ViewerUtils instance for interacting with the parent Viewer

`list_widget`

ROI list widget

`list_widget_tags`

Tags list widget

`vi`

ViewrUtils instance

current_index
Current index (int)

previous_index
Previous index (int)

append (*roi*: Union[mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI, mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI])
Add an ROI instance to the list

clear_ ()
Cleanup of the list

__delitem__ (*key*)
Delete an ROI from the list and cleanup from the viewer, reindex the colors etc.

disconnect_all ()
Disconnect signals from the parent GUI

_reindex_list_widget ()
Reindex ROI list

reindex_colormap ()
Reindex the colors so they sequentially follow the HSV colormap

__getitem__ (*item*) → Union[mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI, mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI]
Get an item (ROI) from the list

set_current_index (*ix*: int)
Set the current index

highlight_roi (*roi*: Union[mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI, mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI])
Highlight an ROI in white, both the spatial visualization and the curve

highlight_curve (*ix*: int)
Highlight the curve corresponding to the ROI at the passed index

set_previous_index ()
Set the previous_index attribute

slot_show_all_checkbox_clicked (*b*: bool)
Show all ROIs in the viewer overlay visualization and curves

_show_graphics_object (*ix*: int)
Show the ROI at the passed index in the viewer overlay visualization

_hide_graphics_object (*ix*: int)
Hide the ROI at the passed index in the viewer overlay visualization

_show_all_graphics_objects ()
Show all ROIs in the viewer overlay visualization

_hide_all_graphics_objects ()
Hide all ROIs in the viewer overlay visualization

plot_manual_roi_regions ()
Plot the ROI curves from the regions of all ManualROI instances in the list

set_pg_roi_plot (*ix*: int)
Plot the ROI curve from the region of the ManualROI instance at the passed index

set_list_widget_tags()
Set the tags list for the ROI at the current index

update_roi_defs_from_configuration()
Update ROI_DEFS in the Tags list from the project configuration

__weakref__
list of weak references to the object (if defined)

46.4 ROI Types

A list of these are held by an instance of *ROIList*

46.4.1 AbstractBaseROI

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI (curve_plot_item:
mesmerize.pyqtgraphCore.view_box:
mesmerize.pyqtgraphCore.state:
Optional[dict])
```

Abstract base class defining an ROI that works with the ROIList and ROI Managers. Inherit from this or BaseROI to make a new ROI class

abstract __init__ (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem, view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox, state: Optional[dict])

Minimum required attributes

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

abstract get_roi_graphics_object () → PyQt5.QtWidgets.QGraphicsObject
Get the QGraphicsObject used for visualization of the spatial localization of the ROI

abstract set_roi_graphics_object (*args, **kwargs)
Set the QGraphicsObject used for visualization of the spatial localization of the ROI

abstract reset_color ()
Reset the color of this ROI back to the original color

abstract set_original_color (color)
Set the original color for this ROI

Parameters color – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

abstract `get_color()` → `numpy.ndarray`

Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RBGA format, [R, G, B, A]

Return type `np.ndarray`

abstract `set_color(color, *args, **kwargs)`

Set the current color of this ROI

Parameters `color` – 1D numpy array of 4 floating point numbers (range 0 - 255) in RBGA format, [R, G, B, A]

abstract `set_text(text: str)`

Not implemented

abstract `set_tag(roi_def: str, tag: str)`

Set a tag for the passed `roi_def`

Parameters

- **roi_def** – The ROI_DEF that should be tagged
- **tag** – The tag to label for the passed ROI_DEF/ROI Type

abstract `get_tag(roi_def)` → `str`

Get the tag that is set to the passed ‘roi_def’

Return type `str`

abstract `get_all_tags()` → `dict`

Get all the tags for all the ROI_DEFS

Return type `dict`

abstract `add_to_viewer()`

Add this ROI to the viewer.

abstract `remove_from_viewer()`

Remove this ROI from the viewer

abstract `to_state()`

Get the current state for this ROI so that it can be restored later

abstract classmethod `from_state(curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem, view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox, state: dict)`

Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

__weakref__

list of weak references to the object (if defined)

46.4.2 BaseROI

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI (curve_plot_item:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems
                                                                    view_box:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems
                                                                    state:
                                                                    Op-
                                                                    tional[dict]
                                                                    =
                                                                    None,
                                                                    meta-
                                                                    data:
                                                                    dict =
                                                                    None)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI`

A base class that is used by ManualROI and CNMFEROI Inherit from this to make a new ROI class

```
__init__ (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
          view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox, state:
          Optional[dict] = None, metadata: dict = None)
```

Instantiate common attributes

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

get_roi_graphics_object () → PyQt5.QtWidgets.QGraphicsObject

Get the QGraphicsObject used for visualization of the spatial localization of the ROI

set_roi_graphics_object (*args, **kwargs)

Set the QGraphicsObject used for visualization of the spatial localization of the ROI

reset_color ()

Reset the color of this ROI back to the original color

set_original_color (color)

Set the original color for this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

get_color ()

Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

Return type np.ndarray

set_color (*color*: Union[numpy.ndarray, str], *args, **kwargs)

Set the current color of this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

set_text (*text*: str)

Not implemented

set_tag (*roi_def*: str, *tag*: str)

Set a tag for the passed roi_def

Parameters

- **roi_def** – The ROI_DEF that should be tagged
- **tag** – The tag to label for the passed ROI_DEF/ROI Type

get_tag (*roi_def*) → str

Get the tag that is set to the passed 'roi_def'

Return type str

get_all_tags () → dict

Get all the tags for all the ROI_DEFS

Return type dict

add_to_viewer ()

Add this ROI to the viewer.

remove_from_viewer ()

Remove this ROI from the viewer

to_state ()

Must be implemented in subclass

classmethod from_state (*curve_plot_item*: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem, *view_box*: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox, *state*: dict)

Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

46.4.3 ManualROI

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI (curve_plot_item:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems.ROI.ROI,
                                                                    roi_graphics_object:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems.QGraphicsObject.QGraphicsObject,
                                                                    view_box:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox,
                                                                    state:
                                                                    Op-
                                                                    tional[dict]
                                                                    =
                                                                    None,
                                                                    spike_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    dfof_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI`

A class manually drawn ROIs

```
__init__ (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
           roi_graphics_object: mesmerize.pyqtgraphCore.graphicsItems.ROI.ROI, view_box: mes-
           mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox, state: Optional[dict] =
           None, spike_data: numpy.ndarray = None, dfof_data: numpy.ndarray = None)
```

property curve_data

tuple of (xs, ys)

Type return

get_roi_graphics_object () → `mesmerize.pyqtgraphCore.graphicsItems.ROI.ROI`

Get the QGraphicsObject used for visualization of the spatial localization of the ROI

set_roi_graphics_object (graphics_object: `mesmerize.pyqtgraphCore.graphicsItems.ROI.ROI`)

Set the QGraphicsObject used for visualization of the spatial localization of the ROI

to_state ()

Must be implemented in subclass

```
classmethod from_state (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
                        view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox,
                        state: dict)
```

Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer

- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

reset_color()

Reset the color of this ROI back to the original color

set_original_color(color)

Set the original color for this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

get_color()

Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

Return type np.ndarray

set_color(color: Union[numpy.ndarray, str], *args, **kwargs)

Set the current color of this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

set_text(text: str)

Not implemented

set_tag(roi_def: str, tag: str)

Set a tag for the passed roi_def

Parameters

- **roi_def** – The ROI_DEF that should be tagged
- **tag** – The tag to label for the passed ROI_DEF/ROI Type

get_tag(roi_def) → str

Get the tag that is set to the passed 'roi_def'

Return type str

get_all_tags() → dict

Get all the tags for all the ROI_DEFS

Return type dict

add_to_viewer()

Add this ROI to the viewer.

remove_from_viewer()

Remove this ROI from the viewer

46.4.4 ScatterROI

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI (curve_plot_item:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphics
                                                                    view_box:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphics
                                                                    state:
                                                                    Op-
                                                                    tional[dict]
                                                                    =
                                                                    None,
                                                                    curve_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    xs:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    ys:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    meta-
                                                                    data:
                                                                    dict
                                                                    =
                                                                    None,
                                                                    spike_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    dfof_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    **kwargs)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI`

A class for unmoveable ROIs drawn using scatter points

```
__init__ (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
          view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox, state:
          Optional[dict] = None, curve_data: numpy.ndarray = None, xs: numpy.ndarray = None,
          ys: numpy.ndarray = None, metadata: dict = None, spike_data: numpy.ndarray = None,
          dfof_data: numpy.ndarray = None, **kwargs)
```

Parameters

- **curve_plot_item** –
- **view_box** –

- **state** –
- **curve_data** – 1D numpy array of y values
- **kwargs** –

set_curve_data (*y_vals: numpy.ndarray*)

Set the curve data

to_state () → *dict*

Must be implemented in subclass

set_roi_graphics_object (*xs: numpy.ndarray, ys: numpy.ndarray*)

Set the QGraphicsObject used for visualization of the spatial localization of the ROI

get_roi_graphics_object () → *mesmerize.pyqtgraphCore.graphicsItems.ScatterPlotItem.ScatterPlotItem*

Get the QGraphicsObject used for visualization of the spatial localization of the ROI

classmethod from_state (*curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem, view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox, state: dict, **kwargs*)

Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

reset_color ()

Reset the color of this ROI back to the original color

set_original_color (*color*)

Set the original color for this ROI

Parameters color – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

get_color ()

Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

Return type *np.ndarray*

set_color (*color: Union[numpy.ndarray, str], *args, **kwargs*)

Set the current color of this ROI

Parameters color – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

set_text (*text: str*)

Not implemented

set_tag (*roi_def: str, tag: str*)

Set a tag for the passed roi_def

Parameters

- **roi_def** – The ROI_DEF that should be tagged

- **tag** – The tag to label for the passed ROI_DEF/ROI Type

get_tag (*roi_def*) → *str*

Get the tag that is set to the passed 'roi_def'

Return type *str*

get_all_tags () → *dict*

Get all the tags for all the ROI_DEFS

Return type *dict*

add_to_viewer ()

Add this ROI to the viewer.

remove_from_viewer ()

Remove this ROI from the viewer

46.4.5 VoICNMF

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types.VoICNMF(curve_plot_item:
    mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox,
    view_box:
    mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox,
    cnmf_idx:
    int =
    None,
    curve_data:
    numpy.ndarray
    =
    None,
    contour:
    dict =
    None,
    state:
    Optional[dict]
    =
    None,
    spike_data:
    numpy.ndarray
    =
    None,
    dfof_data:
    numpy.ndarray
    =
    None,
    metadata:
    dict =
    None,
    zlevel:
    int =
    0)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI`

3D ROI for CNMF data

```
__init__(curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
    view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox, cnmf_idx:
    int = None, curve_data: numpy.ndarray = None, contour: dict = None, state: Optional[dict]
    = None, spike_data: numpy.ndarray = None, dfof_data: numpy.ndarray = None, metadata:
    dict = None, zlevel: int = 0)
```

Parameters

- **curve_plot_item** –
- **view_box** –

- **state** –
- **curve_data** – 1D numpy array of y values
- **kwargs** –

to_state() → dict

Must be implemented in subclass

set_roi_graphics_object()

Set the QGraphicsObject used for visualization of the spatial localization of the ROI

set_zlevel(z: int)

Set the z-level of the ROI to correspond with the z-level of the image.

Different from *setZValue*!!

get_roi_graphics_object() → mesmerize.pyqtgraphCore.graphicsItems.ScatterPlotItem.ScatterPlotItem

Get the QGraphicsObject used for visualization of the spatial localization of the ROI

reset_color()

Reset the color of this ROI back to the original color

set_original_color(color)

Set the original color for this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

get_color()

Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

Return type np.ndarray

set_color(color: Union[numpy.ndarray, str], *args, **kwargs)

Set the current color of this ROI

Parameters **color** – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

set_text(text: str)

Not implemented

set_tag(roi_def: str, tag: str)

Set a tag for the passed roi_def

Parameters

- **roi_def** – The ROI_DEF that should be tagged
- **tag** – The tag to label for the passed ROI_DEF/ROI Type

get_tag(roi_def) → str

Get the tag that is set to the passed 'roi_def'

Return type str

get_all_tags() → dict

Get all the tags for all the ROI_DEFs

Return type dict

add_to_viewer()

Add this ROI to the viewer.

remove_from_viewer()

Remove this ROI from the viewer

classmethod from_state (*curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem*,
view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox,
state: dict, ***kwargs*)

Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer
- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

set_curve_data (*y_vals: numpy.ndarray*)

Set the curve data

46.4.6 CNMFROI

```
class mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI (curve_plot_item:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems.PlotDataItem,
                                                                    view_box:
                                                                    mes-
                                                                    mer-
                                                                    ize.pyqtgraphCore.graphicsItems.ViewBox,
                                                                    cnmf_idx:
                                                                    int =
                                                                    None,
                                                                    curve_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    con-
                                                                    tour:
                                                                    dict =
                                                                    None,
                                                                    state:
                                                                    Op-
                                                                    tional[dict]
                                                                    =
                                                                    None,
                                                                    spike_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    dfof_data:
                                                                    numpy.ndarray
                                                                    =
                                                                    None,
                                                                    meta-
                                                                    data:
                                                                    dict =
                                                                    None,
                                                                    **kwargs)
```

Bases: `mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI`

A class for ROIs imported from CNMF(E) output data

```
__init__ (curve_plot_item: mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem,
          view_box: mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox, cnmf_idx:
          int = None, curve_data: numpy.ndarray = None, contour: dict = None, state: Optional[dict]
          = None, spike_data: numpy.ndarray = None, dfof_data: numpy.ndarray = None, metadata:
          dict = None, **kwargs)
```

Instantiate attributes.

Type curve_data: np.ndarray

Parameters

- **curve_data** – 1D numpy array of y values
- **cnmf_idx** – original index of the ROI from cnmf idx_components

get_roi_graphics_object () → `mesmerize.pyqtgraphCore.graphicsItems.ScatterPlotItem.ScatterPlotItem`
 Get the QGraphicsObject used for visualization of the spatial localization of the ROI

set_roi_graphics_object (*xs*: `numpy.ndarray`, *ys*: `numpy.ndarray`)
 Set the QGraphicsObject used for visualization of the spatial localization of the ROI

reset_color ()
 Reset the color of this ROI back to the original color

set_original_color (*color*)
 Set the original color for this ROI

Parameters *color* – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

get_color ()
 Get the current color of this ROI

Returns 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

Return type `np.ndarray`

set_color (*color*: `Union[numpy.ndarray, str]`, **args*, ***kwargs*)
 Set the current color of this ROI

Parameters *color* – 1D numpy array of 4 floating point numbers (range 0 - 255) in RGBA format, [R, G, B, A]

set_text (*text*: `str`)
 Not implemented

set_tag (*roi_def*: `str`, *tag*: `str`)
 Set a tag for the passed roi_def

Parameters

- **roi_def** – The ROI_DEF that should be tagged
- **tag** – The tag to label for the passed ROI_DEF/ROI Type

get_tag (*roi_def*) → `str`
 Get the tag that is set to the passed 'roi_def'

Return type `str`

get_all_tags () → `dict`
 Get all the tags for all the ROI_DEFS

Return type `dict`

add_to_viewer ()
 Add this ROI to the viewer.

remove_from_viewer ()
 Remove this ROI from the viewer

classmethod from_state (*curve_plot_item*: `mesmerize.pyqtgraphCore.graphicsItems.PlotDataItem.PlotDataItem`, *view_box*: `mesmerize.pyqtgraphCore.graphicsItems.ViewBox.ViewBox.ViewBox`, *state*: `dict`, ***kwargs*)
 Restore this ROI from a state

Parameters

- **curve_plot_item** – The plot item that is used for display the curves in the viewer

- **view_box** – ViewBox containing the image sequence, used for overlaying the ROIs on top of the image
- **state** – ROI state, used for restoring the ROIs. Pass None is not restoring an ROI from a state dict

set_curve_data (*y_vals*: *numpy.ndarray*)

Set the curve data

to_state () → *dict*

Must be implemented in subclass

STIMULUS MAPPING

47.1 ModuleGUI

class mesmerize.viewer.modules.stimulus_mapping.**ModuleGUI** (*parent, viewer*)

property maps

Returns a dictionary of the stimulus maps

47.2 Page

Each Page instance contains the mapping data for one stimulus type

class mesmerize.viewer.modules.stimmap_modules.page.**Page** (*parent, stim_type: str*)

set_data (*dataframe: pandas.core.frame.DataFrame*)

Set the stimulus map

Parameters dataframe – DataFrame with the appropriate rows (see add_row())

get_dataframe () → *pandas.core.frame.DataFrame*

Get the stimulus map as a DataFrame

set_units (*units: str*)

Set the time units

Parameters units – One of ‘frames’ or ‘seconds’

get_units () → *str*

Get the time units

add_row (*pd_series: pandas.core.series.Series = None*)

Add a row to the stimulus map

Parameters pd_series – pandas series containing the following: stimulus name, start, end, and color

Returns

delete_row (*row: Union[mesmerize.viewer.modules.stimmap_modules.row.Row, int]*)

Delete a row from the stimulus map

Parameters row – The Row object to remove or the numerical index of the row

clear()

Clear the stimulus map

47.3 DataFrame Format

Page.set_data() expects a DataFrame in the following format**Columns**

Column	Description
name	Stimulus name
start	Start time of stimulus period
end	End time of stimulus period
color	Color to display in the viewer curve plot

Data types:

Column	Data type
name	str
start	numpy.float64
end	numpy.float64
color	Tuple in RGBA format (int, int, int, int) Each int must be within the 0 - 255 range

Example

name	start	end	color
control	0.0	328.0	(0, 75, 0, 255)
stim_A	328.0	1156.0	(0, 0, 125, 255)
stim_C	1156.0	2987.0	(125, 0, 0, 255)

DATA TYPES USED FOR ANALYSIS

48.1 Transmission

Inherits from BaseTransmission

```
class mesmerize.Transmission(df: pandas.core.frame.DataFrame, history_trace: mesmerize.analysis.data_types.HistoryTrace, proj_path: str = None, last_output: str = None, last_unit: str = None, ROI_DEFS: list = None, STIM_DEFS: list = None, CUSTOM_COLUMNS: list = None, plot_state: dict = None)
```

The transmission object used throughout the flowchart

```
__init__(df: pandas.core.frame.DataFrame, history_trace: mesmerize.analysis.data_types.HistoryTrace, proj_path: str = None, last_output: str = None, last_unit: str = None, ROI_DEFS: list = None, STIM_DEFS: list = None, CUSTOM_COLUMNS: list = None, plot_state: dict = None)
```

Base class for common Transmission functions

Parameters

- **df** (*pd.DataFrame*) – Transmission dataframe
- **history_trace** (*HistoryTrace*) – HistoryTrace object, keeps track of the nodes & node parameters the transmission has been processed through
- **proj_path** (*str*) – Project path, necessary for the datapoint tracer
- **last_output** (*str*) – Last data column that was appended via a node's operation
- **last_unit** (*str*) – Current units of the data. Refers to the units of column in last_output
- **plot_state** (*dict*) – State of a plot, such as data and label columns. Used when saving interactive plots.

Variables

- **df** – DataFrame instance
- **history_trace** – *HistoryTrace* instance
- **last_output** – Name of the DataFrame column that contains data from the most recent node
- **last_unit** – The data units for the data in the column of 'last_output'
- **plot_state** – State of a plot, containing user entered plot parameters. Used for storing interactive plot states.

static empty_df (*transmission*, *addCols*: *list* = *None*) → *pandas.core.frame.DataFrame*

Just a helper method to return an empty DataFrame with the same columns

Parameters

- **transmission** – Transmission object that forms the basis
- **addCols** – list of columns to add

Returns The input transmission with an empty dataframe containing the same columns and any additional columns that were passed

classmethod from_pickle (*path*)

Load Transmission from a pickle.

Parameters **path** – file path, usually ends in .trn

to_pickle (*path*: *str*)

Save Transmission as a pickle. Not recommended for sharing data, use *to_hdf5()*

Parameters **path** – file path, usually ends in .trn

classmethod from_hdf5 (*path*: *str*)

Create Transmission from an hdf5 file. See *HdfTools* for information on the file structure.

Parameters **path** – file path, usually ends in .trn (.ptrn for plots)

to_hdf5 (*path*: *str*)

Save as an hdf5 file. Uses pytables to save the DataFrame, serializes the HistoryTrace using JSON. See *HdfTools*

Parameters **path** – file path, usually ends in .trn

get_proj_path () → *str*

Get the project root dir associated to this Transmission.

Returns Root directory of the project

set_proj_path (*path*: *str*)

Set the project root dir for this transmission.

Used for finding associated project files, for example the Datapoint Tracer uses it to find max and std projections of image sequences.

Parameters **path** – Root directory of the project

to_dict () → *dict*

Package Transmission as a dict, useful for saving to hdf5 or pickle

classmethod from_proj (*proj_path*: *str*, *dataframe*: *pandas.core.frame.DataFrame*,
sub_dataframe_name: *str* = 'root', *dataframe_filter_history*: *dict* =
None)

Parameters

- **proj_path** – root directory of the project
- **dataframe** – Chosen Child DataFrame from the Mesmerize Project
- **sub_dataframe_name** – Name of the sub DataFrame to load
- **dataframe_filter_history** – Filter history of the child dataframe

static _load_files (*proj_path*: *str*, *row*: *pandas.core.series.Series*) → *pandas.core.series.Series*

Loads npz of curve data and pickle files containing metadata using the paths specified in each row of the chosen sub-dataframe of the project

classmethod merge (*transmissions: list*)

Merges a list of Transmissions into one transmission. A single DataFrame is created by simple concatenation. HistoryTrace objects are also merged using HistoryTrace.merge.

Parameters **transmissions** – A list containing Transmission objects to merge

Returns Merged transmission

48.2 BaseTransmission

```
class mesmerize.analysis.data_types.BaseTransmission(df: pandas.core.frame.DataFrame,
                                                    history_trace: mesmerize.analysis.data_types.HistoryTrace,
                                                    proj_path: str = None,
                                                    last_output: str = None,
                                                    last_unit: str = None,
                                                    ROI_DEFS: list = None,
                                                    STIM_DEFS: list = None,
                                                    CUSTOM_COLUMNS: list = None,
                                                    plot_state: dict = None)
```

```
__init__(df: pandas.core.frame.DataFrame, history_trace: mesmerize.analysis.data_types.HistoryTrace, proj_path: str = None, last_output: str = None,
          last_unit: str = None, ROI_DEFS: list = None, STIM_DEFS: list = None, CUSTOM_COLUMNS: list = None, plot_state: dict = None)
```

Base class for common Transmission functions

Parameters

- **df** (*pd.DataFrame*) – Transmission dataframe
- **history_trace** (*HistoryTrace*) – HistoryTrace object, keeps track of the nodes & node parameters the transmission has been processed through
- **proj_path** (*str*) – Project path, necessary for the datapoint tracer
- **last_output** (*str*) – Last data column that was appended via a node's operation
- **last_unit** (*str*) – Current units of the data. Refers to the units of column in last_output
- **plot_state** (*dict*) – State of a plot, such as data and label columns. Used when saving interactive plots.

Variables

- **df** – DataFrame instance
- **history_trace** – *HistoryTrace* instance
- **last_output** – Name of the DataFrame column that contains data from the most recent node
- **last_unit** – The data units for the data in the column of 'last_output'
- **plot_state** – State of a plot, containing user entered plot parameters. Used for storing interactive plot states.

plot_state

If used in a plot, dict containing information about the plot state

to_dict() → dict

Package Transmission as a dict, useful for saving to hdf5 or pickle

to_hdf5(*path: str*)

Save as an hdf5 file. Uses pytables to save the DataFrame, serializes the HistoryTrace using JSON. See [HdfTools](#)

Parameters *path* – file path, usually ends in .trn

classmethod from_hdf5(*path: str*)

Create Transmission from an hdf5 file. See [HdfTools](#) for information on the file structure.

Parameters *path* – file path, usually ends in .trn (.ptrn for plots)

classmethod from_pickle(*path*)

Load Transmission from a pickle.

Parameters *path* – file path, usually ends in .trn

to_pickle(*path: str*)

Save Transmission as a pickle. Not recommended for sharing data, use [to_hdf5\(\)](#)

Parameters *path* – file path, usually ends in .trn

static empty_df(*transmission, addCols: list = None*) → pandas.core.frame.DataFrame

Just a helper method to return an empty DataFrame with the same columns

Parameters

- **transmission** – Transmission object that forms the basis
- **addCols** – list of columns to add

Returns The input transmission with an empty dataframe containing the same columns and any additional columns that were passed

get_proj_path() → str

Get the project root dir associated to this Transmission.

Returns Root directory of the project

set_proj_path(*path: str*)

Set the project root dir for this transmission.

Used for finding associated project files, for example the Datapoint Tracer uses it to find max and std projections of image sequences.

Parameters *path* – Root directory of the project

set_proj_config()

Sets some project config related attributes from the project's config file.

48.3 HistoryTrace

```
class mesmerize.analysis.data_types.HistoryTrace (history: Dict[Union[uuid.UUID, str],  
                                                List[Dict]] = None, data_blocks:  
                                                List[Union[uuid.UUID, str]] =  
                                                None)
```

Structure of a history trace:

A dict with keys that are the block_ids. Each dict value is a list of operation_dicts. Each operation_dict has a single key which is the name of the operation and the value of that key is the operation parameters.

```

{block_id_1: [
    {operation_1:
        {
            param_1: a,
            param_2: b,
            param_n, z
        }
    },
    {operation_2:
        {
            param_1: a,
            param_n, z
        }
    },
    ...
    {operation_n:
        {
            param_n: x
        }
    }
]
block_id_2: <list of operation dicts>,
...
block_id_n: <list of operation dicts>
}

```

The main dict illustrated above should never be worked with directly.

You must use the helper methods of this class to query or add information

```

__init__(history: Dict[Union[uuid.UUID, str], List[Dict]] = None, data_blocks:
        List[Union[uuid.UUID, str]] = None)

```

Parameters

- **history** – Dict containing a data block UUIDs as keys. The values are a list of dicts containing operation parameters.
- **data_blocks** – List of data block UUIDs

Variables

- **_history** – The dict of the actual data, as illustrated above. Should not be accessed directly. Use the `history` property or call `get_all_data_blocks_history()`.
- **_data_blocks** – List of all data blocks. Should not be accessed directly, use the `data_blocks` property instead.

property data_blocks

List of UUIDs that allow you to pin down the history of specific rows of the dataframe to their history as stored in the history trace data structure (self.history)

property history

The analysis log that is stored in the structure outlined in the doc string

create_data_block (*dataframe:* *pandas.core.frame.DataFrame*) → Tuple[pandas.core.frame.DataFrame, uuid.UUID]

Creates a new UUID, assigns it to the input dataframe by setting the UUID in the `_BLOCK_` column

Parameters **dataframe** – Assigns a block ID to this entire DataFrame.

_add_data_block (*data_block_id:* *uuid.UUID*)

Adds new datablock UUID to the list of datablocks in this instance. Throws exception if UUID already exists.

add_operation (*data_block_id:* Union[*uuid.UUID*, *str*], *operation:* *str*, *parameters:* *dict*)

Add a single operation, that is usually performed by a node, to the history trace. Added to all or specific datablock(s), depending on which datablock(s) the node performed the operation on

Parameters

- **data_block_id** – data_block_id to log the operation on to. either a UUID or ‘all’ to append the operation to all data blocks
- **operation** – name of the operation, usually the same as the name of the node in all lowercase
- **parameters** – operation parameters.

get_data_block_history (*data_block_id:* Union[*str*, *uuid.UUID*], *copy:* *bool* = *False*) → List[dict]

Get the full history trace of a single data block.

Use copy=False if you want to modify the history trace of the data block.

Parameters

- **data_block_id** (Union[*str*, *UUID*]) – data block ID
- **copy** (*bool*) – If true, returns a deepcopy

Returns data block history

Return type List[dict]

get_all_data_blocks_history () → dict

Returns history trace of all datablocks

get_operations_list (*data_block_id:* Union[*uuid.UUID*, *str*]) → list

Returns just a simple list of operations in the order that they were performed on the given datablock. To get the operations along with their parameters call `get_data_block_history()`

get_operation_params (*data_block_id:* Union[*uuid.UUID*, *str*], *operation:* *str*) → dict

Get the parameters dict for a specific operation that was performed on a specific data block

check_operation_exists (*data_block_id:* *uuid.UUID*, *operation:* *str*) → bool

Check if a specific operation was performed on a specific datablock

static _to_uuid (*u:* Union[*str*, *uuid.UUID*]) → *uuid.UUID*

If argument ‘u’ is type <str> that can be formatted as a UUID, return it as UUID type. If argument ‘u’ is a UUID, just return it.

to_dict () → dict

Package the HistoryTrace instance as a dict. Converts all UUIDs to <str> representation for JSON compatibility.

static from_dict (d: dict) → dict

Format a dict stored using HistoryTrace.to_dict so that it can be used to create a HistoryTrace instance. Converts all the <str> representations of UUID back to <uuid.UUID> types.

Parameters **d** – dict containing appropriate ‘history’ and ‘datablocks’ keys. Must be packaged by HistoryTrace.to_dict()

Returns dict formatted so that it can be used to instantiate a HistoryTrace instance recapitulating the HistoryTrace it was packaged from.

to_json (path: str)

Save HistoryTrace to a JSON file.

Parameters **path** – file path, usually ends with .json

classmethod from_json (path: str)

Instantiate HistoryTrace from JSON file (that was saved using HistoryTrace.to_json)

Parameters **path** – file path, usually ends with .json

to_pickle (path: str)

Dump this instance to a pickle

Parameters **path** – file path

classmethod from_pickle (path: str)

Load HistoryTrace that was pickled

Parameters **path** – file path

classmethod merge (history_traces: list)

Merge a list of HistoryTrace instances into one HistoryTrace instance. Useful when merging Transmission objects.

Parameters **history_traces** – list of HistoryTrace instances

draw_graph (data_block_id: Union[str, uuid.UUID], **kwargs) → str

Draw graph of a data block. kwargs are passed to *mesmerize.plotting.utils.draw_graph*

Parameters **data_block_id** (Union[str, UUID]) – data block ID from which to get history to draw in a graph

Returns file path to the graph pdf file

Return type str

static clean_history_trace (db_history: list) → list

Cleans up excessive data such as frequencies linspaces and linkage matrices so the graph is viewable.

Parameters **db_history** – data block history

Returns data block history with excessive params removed

__weakref__

list of weak references to the object (if defined)

ANALYSIS

Analysis helper functions

49.1 Utils

`mesmerize.analysis.utils.get_array_size` (*transmission:* `mesmerize.analysis.data_types.Transmission`,
data_column: `str`) \rightarrow `int`

Returns the size of the 1D arrays in the specified data column. Throws an exception if they do not match

Parameters

- **transmission** (`Transmission`) – Desired Transmission
- **data_column** (`str`) – Data column of the Transmission from which to retrieve the size

Returns Size of the 1D arrays of the specified data column

Return type `int`

`mesmerize.analysis.utils.get_frequency_linspace` (*transmission:* `mesmerize.analysis.data_types.Transmission`)
 \rightarrow `Tuple[numpy.ndarray, float]`

Get the frequency linspace.

Throws an exception if all datablocks do not have the same linspace & Nyquist frequencies

Parameters **transmission** – Transmission containing data from which to get frequency linspace

Returns tuple: (frequency linspace as a 1D numpy array, nyquist frequency)

Return type `Tuple[np.ndarray, float]`

`mesmerize.analysis.utils.get_proportions` (*xs:* `Union[pandas.core.series.Series, numpy.ndarray, list]`, *ys:* `Union[pandas.core.series.Series, numpy.ndarray]`, *xs_name:* `str` = 'xs', *ys_name:* `str` = 'ys', *swap:* `bool` = False, *percentages:* `bool` = True) \rightarrow `pandas.core.frame.DataFrame`

Get the proportions of xs vs ys.

xs & ys are categorical data.

Parameters

- **xs** (`Union[pd.Series, np.ndarray]`) – data plotted on the x axis

- **ys** (*Union*[*pd.Series*, *np.ndarray*]) – proportions of unique elements in ys are calculated per xs
- **xs_name** (*str*) – name for the xs data, useful for labeling the axis in plots
- **ys_name** (*str*) – name for the ys data, useful for labeling the axis in plots
- **swap** (*bool*) – swap x and y

Returns DataFrame that can be plotted in a proportions bar graph

Return type *pd.DataFrame*

mesmerize.analysis.utils.get_sampling_rate (*transmission:* *mesmerize.analysis.data_types.Transmission*, *tolerance: Optional[float] = 0.1*) → *float*

Returns the mean sampling rate of all data in a *Transmission* if it is within the specified tolerance. Otherwise throws an exception.

Parameters

- **transmission** (*Transmission*) – *Transmission* object of the data from which sampling rate is obtained.
- **tolerance** (*float*) – Maximum tolerance (in Hertz) of sampling rate variation between different samples

Returns The mean sampling rate of all data in the *Transmission*

Return type *float*

mesmerize.analysis.utils.organize_dataframe_columns (*columns: Iterable[str]*) → *Tuple[List[str], List[str], List[str]]*

Organizes *DataFrame* columns into data column, categorical label columns, and uuid columns.

Parameters **columns** – All *DataFrame* columns

Returns (*data_columns*, *categorical_columns*, *uuid_columns*)

Return type *Tuple[List[str], List[str], List[str]]*

mesmerize.analysis.utils.pad_arrays (*a: numpy.ndarray*, *method: str = 'random'*, *output_size: int = None*, *mode: str = 'minimum'*, *constant: Any = None*) → *numpy.ndarray*

Pad all the input arrays so that are of the same length. The length is determined by the largest input array. The padding value for each input array is the minimum value in that array.

Padding for each input array is either done after the array's last index to fill up to the length of the largest input array (method 'fill-size') or the padding is randomly flanked to the input array (method 'random') for easier visualization.

Parameters

- **a** (*np.ndarray*) – 1D array where each element is a 1D array
- **method** (*str*) – one of 'fill-size' or 'random', see docstring for details
- **output_size** – not used
- **mode** (*str*) – one of either 'constant' or 'minimum'. If 'minimum' the min value of the array is used as the padding value. If 'constant' the values passed to the "constant" argument is used as the padding value.
- **constant** (*Any*) – padding value if 'mode' is set to 'constant'

Returns Arrays padded according to the chosen method. 2D array of shape [*n_arrays*, size of largest input array]

Return type np.ndarray

49.2 Cross correlation

49.2.1 functions

Helper functions. Uses `tslearn.cyc`

`mesmerize.analysis.math.cross_correlation.ncc_c` (*x*: `numpy.ndarray`, *y*: `numpy.ndarray`)
→ `numpy.ndarray`

Must pass 1D array to both *x* and *y*

Parameters

- **x** – Input array [*x*₁, *x*₂, *x*₃, ... *x*_n]
- **y** – Input array [*y*₁, *y*₂, *y*₃, ... *y*_n]

Returns Returns the normalized cross correlation function (as an array) of the two input vector arguments “*x*” and “*y*”

Return type np.ndarray

`mesmerize.analysis.math.cross_correlation.get_omega` (*x*: `numpy.ndarray` = *None*,
y: `numpy.ndarray` = *None*, *cc*:
`numpy.ndarray` = *None*) → `int`

Must pass a 1D array to either both “*x*” and “*y*” or a cross-correlation function (as an array) to “*cc*”

Parameters

- **x** – Input array [*x*₁, *x*₂, *x*₃, ... *x*_n]
- **y** – Input array [*y*₁, *y*₂, *y*₃, ... *y*_n]
- **cc** – cross-correlation function represented as an array [*c*₁, *c*₂, *c*₃, ... *c*_n]

Returns index (*x*-axis position) of the global maxima of the cross-correlation function

Return type np.ndarray

`mesmerize.analysis.math.cross_correlation.get_lag` (*x*: `numpy.ndarray` = *None*, *y*:
`numpy.ndarray` = *None*, *cc*:
`numpy.ndarray` = *None*) → `float`

Must pass a 1D array to either both “*x*” and “*y*” or a cross-correlation function (as an array) to “*cc*”

Parameters

- **x** – Input array [*x*₁, *x*₂, *x*₃, ... *x*_n]
- **y** – Input array [*y*₁, *y*₂, *y*₃, ... *y*_n]
- **cc** – cross-correlation function represented as a array [*c*₁, *c*₂, *c*₃, ... *c*_n]

Returns Position of the maxima of the cross-correlation function with respect to middle point of the cross-correlation function

Return type np.ndarray

`mesmerize.analysis.math.cross_correlation.get_epsilon` (*x*: `numpy.ndarray` = *None*, *y*:
`numpy.ndarray` = *None*, *cc*:
`numpy.ndarray` = *None*) → `float`

Must pass a 1D vector to either both “*x*” and “*y*” or a cross-correlation function to “*cc*”

Parameters

- **x** – Input array [x1, x2, x3, ... xn]
- **y** – Input array [y2, y2, x3, ... yn]
- **cc** – cross-correlation function represented as an array [c1, c2, c3, ... cn]

Returns Magnitude of the global maxima of the cross-correlation function

Return type np.ndarray

```
mesmerize.analysis.math.cross_correlation.get_lag_matrix (curves:  numpy.ndarray
                                                           =      None,    ccs:
                                                           numpy.ndarray = None)
                                                           → numpy.ndarray
```

Get a 2D matrix of lags. Can pass either a 2D array of 1D curves or cross-correlations

Parameters

- **curves** – 2D array of 1D curves
- **ccs** – 2D array of 1D cross-correlation functions represented by arrays

Returns 2D matrix of lag values, shape is [n_curves, n_curves]

Return type np.ndarray

```
mesmerize.analysis.math.cross_correlation.get_epsilon_matrix (curves:
                                                             numpy.ndarray
                                                             =      None,    ccs:
                                                             numpy.ndarray
                                                             =      None)    →
                                                             numpy.ndarray
```

Get a 2D matrix of maximas. Can pass either a 2D array of 1D curves or cross-correlations

Parameters

- **curves** – 2D array of 1D curves
- **ccs** – 2D array of 1D cross-correlation functions represented by arrays

Returns 2D matrix of maxima values, shape is [n_curves, n_curves]

Return type np.ndarray

```
mesmerize.analysis.math.cross_correlation.compute_cc_data (curves:
                                                           numpy.ndarray)
                                                           →      mesmer-
                                                           ize.analysis.math.cross_correlation.CC_Data
```

Compute cross-correlation data (cc functions, lag and maxima matrices)

Parameters **curves** – input curves as a 2D array, shape is [n_samples, curve_size]

Returns cross correlation data for the input curves as a CC_Data instance

Return type [*CC_Data*](#)

```
mesmerize.analysis.math.cross_correlation.compute_ccs (a:      numpy.ndarray)    →
                                                         numpy.ndarray
```

Compute cross-correlations between all 1D curves in a 2D input array

Parameters **a** – 2D input array of 1D curves, shape is [n_samples, curve_size]

Return type np.ndarray

49.2.2 CC_Data

Data container

Warning: All arguments MUST be `numpy.ndarray` type for `CC_Data` for the save to be saveable as an hdf5 file. Set `numpy.unicode` as the **dtype** for the `curve_uuids` and `labels` arrays. If the **dtype** is 'O' (object) the `to_hdf5()` method will fail.

```
class mesmerize.analysis.cross_correlation.CC_Data(ccs:  numpy.ndarray = None,
                                                    lag_matrix:  numpy.ndarray
                                                    = None,  epsilon_matrix:
numpy.ndarray      = None,
curve_uuids:  numpy.ndarray
= None, labels: numpy.ndarray =
None)
```

```
__init__(ccs:  numpy.ndarray = None, lag_matrix:  numpy.ndarray = None, epsilon_matrix:
numpy.ndarray = None, curve_uuids: numpy.ndarray = None, labels: numpy.ndarray =
None)
```

Object for organizing cross-correlation data

types must be `numpy.ndarray` to be compatible with hdf5

Parameters

- **ccs** (`np.ndarray`) – array of cross-correlation functions, shape: `[n_curves, n_curves, func_length]`
- **lag_matrix** (`np.ndarray`) – the lag matrix, shape: `[n_curves, n_curves]`
- **epsilon_matrix** (`np.ndarray`) – the maxima matrix, shape: `[n_curves, n_curves]`
- **curve_uuids** (`np.ndarray`) – uuids (str representation) for each of the curves, length: `n_curves`
- **labels** (`np.ndarray`) – labels for each curve, length: `n_curves`

ccs

array of cross-correlation functions

lag_matrix

lag matrix

epsilon_matrix

maxima matrix

curve_uuids

uuids for each curve

labels

labels for each curve

```
get_threshold_matrix(matrix_type: str, lag_thr: float, max_thr: float, lag_thr_abs: bool =
True) → numpy.ndarray
```

Get lag or maxima matrix with thresholds applied. Values outside the threshold are set to NaN

Parameters

- **matrix_type** – one of 'lag' or 'maxima'

- **lag_thr** – lag threshold
- **max_thr** – maxima threshold
- **lag_thr_abs** – threshold with the absolute value of lag

Returns the requested matrix with the thresholds applied to it.

Return type np.ndarray

classmethod from_dict (*d: dict*)
Load data from a dict

to_hdf5 (*path: str*)
Save as an HDF5 file

Parameters path – path to save the hdf5 file to, file must not exist.

classmethod from_hdf5 (*path: str*)
Load cross-correlation data from an hdf5 file

Parameters path – path to the hdf5 file

49.3 Clustering metrics

`mesmerize.analysis.clustering_metrics.get_centerlike` (*cluster_members:*
numpy.ndarray, metric: Optional[Union[str, callable]]
= None, dist_matrix: Optional[numpy.ndarray]
= None) → Tuple[numpy.ndarray, int]

Finds the 1D time-series within a cluster that is the most centerlike

Parameters

- **cluster_members** – 2D numpy array in the form [n_samples, 1D time_series]
- **metric** – Metric to use for pairwise distance calculation, simply passed to `sklearn.metrics.pairwise_distances`
- **dist_matrix** – Distance matrix of the cluster members

Returns The cluster member which is most centerlike, and its index in the `cluster_members` array

`mesmerize.analysis.clustering_metrics.get_cluster_radius` (*cluster_members:*
numpy.ndarray, metric: Optional[Union[str, callable]] = None,
dist_matrix: Optional[numpy.ndarray] = None,
centerlike_index: Optional[int] = None)
→ float

Returns the cluster radius according to chosen distance metric

Parameters

- **cluster_members** – 2D numpy array in the form [n_samples, 1D time_series]
- **metric** – Metric to use for pairwise distance calculation, simply passed to `sklearn.metrics.pairwise_distances`

- **dist_matrix** – Distance matrix of the cluster members
- **centerlike_index** – Index of the centerlike cluster member within the cluster_members array

Returns The cluster radius, average between the most centerlike member and all other members

```
mesmerize.analysis.clustering_metrics.davies_bouldin_score(data: numpy.ndarray,  
                                                           cluster_labels:  
                                                           numpy.ndarray,  
                                                           metric: Union[str,  
                                                           callable]) → float
```

Adopted from sklearn.metrics.davies_bouldin_score to use any distance metric

Parameters

- **data** – Data that was used for clustering, [n_samples, 1D time_series]
- **metric** – Metric to use for pairwise distance calculation, simply passed to sklearn.metrics.pairwise_distances
- **cluster_labels** – Cluster labels

Returns Davies Bouldin Score using EMD

50.1 Data

```
class mesmerize.pyqtgraphCore.flowchart.library.Data.DropNa (*args, **kwargs)
    Drop NaNs from the DataFrame

class mesmerize.pyqtgraphCore.flowchart.library.Data.LoadFile (name)
    Load Transmission data object from pickled file

class mesmerize.pyqtgraphCore.flowchart.library.Data.LoadProjDF (name)
    Load raw project DataFrames as Transmission

class mesmerize.pyqtgraphCore.flowchart.library.Data.Merge (name)
    Merge transmissions

class mesmerize.pyqtgraphCore.flowchart.library.Data.NormRaw (name,    ui=None,
                                                                terminals=None,
                                                                **kwargs)
    Normalize between raw min and max values.

class mesmerize.pyqtgraphCore.flowchart.library.Data.PadArrays (name, ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
    Pad 1-D numpy arrays in a particular column

class mesmerize.pyqtgraphCore.flowchart.library.Data.Save (name)
    Save Transmission data object

class mesmerize.pyqtgraphCore.flowchart.library.Data.SelectColumns (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)

class mesmerize.pyqtgraphCore.flowchart.library.Data.SelectRows (name,
                                                                ui=None, ter-
                                                                minals=None,
                                                                **kwargs)

class mesmerize.pyqtgraphCore.flowchart.library.Data.SpliceArrays (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
    Splice 1-D numpy arrays in a particular column.
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Data.TextFilter (name,  
                                                                ui=None, terminals=None,  
                                                                **kwargs)
```

Simple string filtering in a specified column

```
class mesmerize.pyqtgraphCore.flowchart.library.Data.ViewHistory (*args,  
                                                                **kwargs)
```

View History Trace of the input Transmission

```
class mesmerize.pyqtgraphCore.flowchart.library.Data.ViewTransmission (name)  
    View transmission using the spyder object editor
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Data.iloc (name, ui=None, terminals=None, **kwargs)
```

Pass only one or multiple DataFrame Indices

50.2 Display

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.AnalysisGraph (name)  
    Graph of the analysis log
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.BeeswarmPlots (name)  
    Beeswarm and Violin plots
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.CrossCorr (name)  
    Cross Correlation
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.FrequencyDomainMagnitude (name,  
                                                                ui=None,  
                                                                terminals=None,  
                                                                **kwargs)
```

Plot Frequency vs. Frequency Domain Magnitude

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.Heatmap (name)  
    Stack 1-D arrays and plot visually like a heatmap
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.Plot (name)  
    Plot curves and/or scatter points
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.Proportions (name)  
    Plot proportions of one categorical column vs another
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.ScatterPlot (name)  
    Scatter Plot, useful for visualizing transformed data and clusters
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.SpaceMap (name)  
    Visualize spatial maps of a categorical variable
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Display.Timeseries (name)  
    Seaborn Timeseries plot
```

50.3 Signal

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.ButterWorth (name,
                                                                    ui=None,
                                                                    termi-
                                                                    nals=None,
                                                                    **kwargs)
```

Butterworth Filter

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.Normalize (name,
                                                                    ui=None,
                                                                    termi-
                                                                    nals=None,
                                                                    **kwargs)
```

Normalize a column containing 1-D arrays such that values in each array are normalized between 0 and 1

Output Column -> Input Column

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.PeakDetect (name,
                                                                    **kwargs)
```

Detect peaks & bases by finding local maxima & minima. Use this after the Derivative Filter

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.PeakFeatures (*args,
                                                                    **kwargs)
```

Extract peak features after peak detection

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.PowerSpectralDensity (name,
                                                                    ui=None,
                                                                    ter-
                                                                    mi-
                                                                    nals=None,
                                                                    **kwargs)
```

Return the Power Spectral Density of a curve.

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.RFFT (name,      ui=None,
                                                                    terminals=None,
                                                                    **kwargs)
```

Uses fftpack.rfft, 'Discrete Fourier transform of a real sequence.

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.rfft.html#scipy.fftpack.rfft>

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.Resample (name,
                                                                    ui=None, ter-
                                                                    minals=None,
                                                                    **kwargs)
```

Resample 1D data, uses scipy.signal.resample. "Rs" is the new sampling rate in "Tu" units of time. If "Tu" = 1, then Rs is the new sampling rate in Hertz.

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.SavitzkyGolay (name,
                                                                    ui=None,
                                                                    termi-
                                                                    nals=None,
                                                                    **kwargs)
```

Savitzky-Golay filter.

```
class mesmerize.pyqtgraphCore.flowchart.library.Signal.ScalerMeanVariance (name,  
                                                                    ui=None,  
                                                                    ter-  
                                                                    mi-  
                                                                    nals=None,  
                                                                    **kwargs)  
  
    Scaler for time series. Scales time series so that their mean (resp. standard deviation) in each dimension is mu  
(resp. std).  
  
    See https://tslearn.readthedocs.io/en/latest/gen\_modules/preprocessing/tslearn.preprocessing.  
TimeSeriesScalerMeanVariance.html#tslearn.preprocessing.TimeSeriesScalerMeanVariance  
  
class mesmerize.pyqtgraphCore.flowchart.library.Signal.SigmaMAD (name,  
                                                                    ui=None, ter-  
                                                                    minals=None,  
                                                                    **kwargs)  
  
class mesmerize.pyqtgraphCore.flowchart.library.Signal.iRFFT (name,    ui=None,  
                                                                    terminals=None,  
                                                                    **kwargs)  
  
    Uses fftpack.irfft, 'Return inverse discrete Fourier transform of real sequence.'  
  
    https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.irfft.html#scipy.fftpack.irfft  
  
    Input must have an _RFFT column from the RFFT node.
```

50.4 Math

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.AbsoluteValue (name,  
                                                                    ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)  
  
    Performs numpy.abs(<input>). Returns root-mean-square value if <input> is complex  
  
class mesmerize.pyqtgraphCore.flowchart.library.Math.ArgGroupStat (name,  
                                                                    ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)  
  
    Group by a certain column and return value of another column based on a data column statistic  
  
class mesmerize.pyqtgraphCore.flowchart.library.Math.ArrayStats (name,  
                                                                    ui=None, ter-  
                                                                    minals=None,  
                                                                    **kwargs)  
  
    Perform various statistical functions  
  
class mesmerize.pyqtgraphCore.flowchart.library.Math.Derivative (name,  
                                                                    ui=None, ter-  
                                                                    minals=None,  
                                                                    **kwargs)  
  
    Return the Derivative of a curve.  
  
class mesmerize.pyqtgraphCore.flowchart.library.Math.Integrate (name, ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)
```

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.LinRegress (name,
                                                                ui=None, ter-
                                                                minals=None,
                                                                **kwargs)
```

Linear Regression

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.LogTransform (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
```

Can perform various log transforms

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.TVDiff (name,      ui=None,
                                                                terminals=None,
                                                                **kwargs)
```

Total Variation Regularized Numerical Differentiation, Chartrand 2011 method

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.XpowerY (name,      ui=None,
                                                                terminals=None,
                                                                **kwargs)
```

Raise each element of arrays in data column to the exponent Y

```
class mesmerize.pyqtgraphCore.flowchart.library.Math.ZScore (name,      ui=None,
                                                                terminals=None,
                                                                **kwargs)
```

Z-Score the input data. Uses scipy.stats.zscore. Computes over sub-DataFrames that are created according to the “group_by” column parameter

50.5 Biology

```
class mesmerize.pyqtgraphCore.flowchart.library.Biology.ExtractStim (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
```

Extract portions of curves according to stimulus maps

```
class mesmerize.pyqtgraphCore.flowchart.library.Biology.ManualDFoF (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
```

Set Fo for dF/Fo using a particular time period. Useful for looking at stimulus responses

```
class mesmerize.pyqtgraphCore.flowchart.library.Biology.StaticDFoFo (name,
                                                                ui=None,
                                                                termi-
                                                                nals=None,
                                                                **kwargs)
```

Perform $(F - F_o) / F_o$ without a rolling window to find F_o

```
class mesmerize.pyqtgraphCore.flowchart.library.Biology.StimTuning (name)
    Get stimulus tuning curves
```

50.6 Clustering

```
class mesmerize.pyqtgraphCore.flowchart.library.Clustering.KMeans (name,  
                                                                    ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)  
    KMeans clustering https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html  
    Output column -> KMEANS_CLUSTER_<data_column>  
  
class mesmerize.pyqtgraphCore.flowchart.library.Clustering.KShape (name)  
    k-Shape clustering
```

50.7 Hierarchical

```
class mesmerize.pyqtgraphCore.flowchart.library.Hierarchical.FCluster (name,  
                                                                    **kwargs)  
    Basically scipy.cluster.hierarchy.fcluster. Form flat clusters from the hierarchical clustering defined by the given  
    linkage matrix.  
  
class mesmerize.pyqtgraphCore.flowchart.library.Hierarchical.Inconsistent (name)  
    Calculate inconsistency statistics on a linkage matrix. Returns inconsistency matrix  
  
class mesmerize.pyqtgraphCore.flowchart.library.Hierarchical.Linkage (name,  
                                                                    ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)  
    Basically scipy.cluster.hierarchy.linkage Compute a linkage matrix for Hierarchical clustering  
  
class mesmerize.pyqtgraphCore.flowchart.library.Hierarchical.MaxIncStat (name,  
                                                                    **kwargs)  
    Return the maximum statistic for each non-singleton cluster and its children.  
  
class mesmerize.pyqtgraphCore.flowchart.library.Hierarchical.MaxInconsistent (name,  
                                                                    **kwargs)  
    Return the maximum inconsistency coefficient for each non-singleton cluster and its children.
```

50.8 Transform

```
class mesmerize.pyqtgraphCore.flowchart.library.Transform.LDA (name, **kwargs)  
    Linear Discriminant Analysis, uses sklearn  
  
class mesmerize.pyqtgraphCore.flowchart.library.Transform.Manifold (name,  
                                                                    ui=None,  
                                                                    termi-  
                                                                    nals=None,  
                                                                    **kwargs)  
    Manifold learning  
  
class mesmerize.pyqtgraphCore.flowchart.library.Transform.PCA (name, ui=None,  
                                                                    terminals=None,  
                                                                    **kwargs)  
    Principal component analysis. Uses sklearn.decomposition.PCA
```

50.9 CtrlNode

Base for all nodes

```
class mesmerize.pyqtgraphCore.flowchart.library.common.CtrlNode(name,
                                                                    ui=None, terminals=None,
                                                                    **kwargs)
```

Abstract class for nodes with auto-generated control UI

ctrlWidget ()

Return this Node's control widget.

By default, Nodes have no control widget. Subclasses may reimplement this method to provide a custom widget. This method is called by Flowcharts when they are constructing their Node list.

process (***kwargs*)

Process data through this node. This method is called any time the flowchart wants the node to process data. It will be called with one keyword argument corresponding to each input terminal, and must return a dict mapping the name of each output terminal to its new value.

This method is also called with a 'display' keyword argument, which indicates whether the node should update its display (if it implements any) while processing this data. This is primarily used to disable expensive display operations during batch processing.

saveState ()

Return a dictionary representing the current state of this node (excluding input / output values). This is used for saving/reloading flowcharts. The default implementation returns this Node's position, bypass state, and information about each of its terminals.

Subclasses may want to extend this method, adding extra keys to the returned dict.

restoreState (*state*)

Restore the state of this node from a structure previously generated by saveState().

PLOTTING UTILS

A few useful helper functions

`mesmerize.plotting.utils.get_colormap(labels: iter, cmap: str, **kwargs) → collections.OrderedDict`

Get a dict for mapping labels onto colors

Any kwargs are passed to `auto_colormap()`

Parameters

- **labels** – labels for creating a colormap. Order is maintained if it is a list of unique elements.
- **cmap** – name of colormap

Returns dict of labels as keys and colors as values

`mesmerize.plotting.utils.map_labels_to_colors(labels: iter, cmap: str, **kwargs) → list`

Map labels onto colors according to chosen colormap

Any kwargs are passed to `auto_colormap()`

Parameters

- **labels** – labels for mapping onto a colormap
- **cmap** – name of colormap

Returns list of colors mapped onto the labels

`mesmerize.plotting.utils.auto_colormap(n_colors: int, cmap: str = 'hsv', output: str = 'mpl', spacing: str = 'uniform', alpha: float = 1.0) → List[Union[PyQt5.QtGui.QColor, numpy.ndarray]]`

If non-qualitative map: returns list of colors evenly spread through the chosen colormap. If qualitative map: returns subsequent colors from the chosen colormap

Parameters

- **n_colors** – Numbers of colors to return
- **cmap** – name of colormap
- **output** – option: 'mpl' returns RGBA values between 0-1 which matplotlib likes, option: 'pyqt' returns `QtGui.QColor` instances that correspond to the RGBA values
- **spacing** – option: 'uniform' returns evenly spaced colors across the entire cmap range, option: 'subsequent' returns subsequent colors from the cmap
- **alpha** – alpha level, 0.0 - 1.0

Returns List of colors as either `QColor` or numpy array with length `n_colors`

class mesmerize.plotting.utils.WidgetRegistry

Register widgets to conveniently store and restore their states

register (*widget*: *PyQt5.QtWidgets.QWidget*, *setter*: *callable*, *getter*: *callable*, *name*: *str*)

Register a widget. The *setter* and *getter* methods must be interoperable

Parameters

- **widget** (*QtWidgets.QWidget*) – widget to register
- **setter** (*callable*) – widget’s method to use for setting its value
- **getter** (*callable*) – widget’s method to use for getting its value. This value must be settable through the specified “setter” method
- **name** (*str*) – a name for this widget

get_state () → *dict*

Get a dict of states for all registered widgets

set_state (*state*: *dict*)

Set all registered widgets from a dict

PLOT BASES

52.1 AbstractBasePlotWidget

```
class mesmerize.plotting.widgets.base._AbstractBasePlotWidget
```

```
    abstract property transmission
```

The input transmission

Return type *Transmission*

```
    abstract set_input (transmission: mesmerize.analysis.data_types.Transmission)
```

Set the input Transmission with data to plot

Parameters **transmission** – Input transmission

```
    abstract update_plot (*args, **kwargs)
```

Method that must be used for updating the plot

```
    abstract get_plot_opts () → dict
```

Package all necessary plot parameters that in combination with the transmission property are sufficient to restore the plot

```
    abstract set_plot_opts (opts: dict)
```

Set plot parameters from a dict in the format returned by get_plot_opts()

```
    abstract save_plot (*args)
```

Package plot data and plot parameters and save to a file. Must contain all the information that is necessary to restore the plot

```
    abstract open_plot (ptrn_path: str, proj_path: str) → Optional[Tuple[str, str]]
```

Open a plot file and restore the plot

52.2 BasePlotWidget

Inherit from this to create interactive plots that can be saved and restored.

```
class mesmerize.plotting.widgets.base.BasePlotWidget
```

Bases: *mesmerize.plotting.widgets.base._AbstractBasePlotWidget*

Base for plot widgets.

Subclasses must define the class attribute “drop_opts” which is used to store a list of JSON incompatible keys returned by the get_plot_opts() method

__init__ ()

Initialize self. See help(type(self)) for accurate signature.

block_signals_list

List of QObjects included in dynamic signal blocking. Used for storing plot parameter widgets so that changing all of them quickly (like when restoring a plot) doesn't cause the plot to constantly update.

property transmission

The input transmission

Return type *Transmission*

set_input (*transmission: mesmerize.analysis.data_types.Transmission*)

Set the input Transmission with data to plot

Parameters transmission – Input transmission

fill_control_widget (*data_columns: list, categorical_columns: list, uuid_columns: list*)

Method for filling the control widget(s) when inputs are set. Must be implemented in subclass

update_plot (*args, **kwargs)

Must be implemented in subclass

get_plot_opts (*drop: bool*) → dict

Must be implemented in subclass

set_plot_opts (*opts: dict*)

Must be implemented in subclass

classmethod signal_blocker (*func*)

Use as a decorator. Block Qt signals from all QObjects instances in the block_signals_list

save_plot_dialog (*path, *args*)

Plot save dialog

save_plot (*path*)

Save the plot as a Transmission in an HDF5 file. Plot parameters are stored as a JSON string within the HDF5 file. See Transmission.to_hdf5

Parameters path – Path to save the file to. For easy identification use “.ptrn” extension.

open_plot_dialog (*filepath, dirpath, *args, **kwargs*)

Open plot dialog

open_plot (*ptrn_path: str, proj_path: str*) → Optional[Tuple[str, str]]

Open a plot saved by the save_plot() method

Parameters

- **ptrn_path** – Path to the HDF5 Transmission file. By convention file extension is “.ptrn”
- **proj_path** – Project path of the associated plot data.

DATAPPOINT TRACER

User guide

class mesmerize.plotting.DatapointTracerWidget

set_widget (datapoint_uuid: *uuid.UUID*, data_column_curve: *str*, row: *pandas.core.series.Series*, proj_path: *str*, history_trace: *Optional[list]* = None, peak_ix: *Optional[int]* = None, tstart: *Optional[int]* = None, tend: *Optional[int]* = None, roi_color: *Optional[Union[str, float, int, tuple]]* = 'ff0000', clear_linear_regions: *bool* = True)

Set the widget from the datapoint.

Parameters

- **datapoint_uuid** – appropriate UUID for the datapoint (such as `uuid_curve` or `_pfeature_uuid`)
- **data_column_curve** – data column containing an array to plot
- **row** – DataFrame row that corresponds to the datapoint
- **proj_path** – root dir of the project the datapoint comes from, used for finding max & std projections
- **history_trace** – history trace of the datablock the datapoint comes from
- **peak_ix** – Deprecated
- **tstart** – lower bounds for drawing *LinearRegionItem*
- **tend** – upper bounds for drawing *LinearRegionItem*
- **roi_color** – color for drawing the spatial bounds of the ROI

set_image (*projection: str*)

Set either the max or std projection image

Parameters **projection** – one of either ‘max’ or ‘std’

open_in_viewer ()

Open the parent Sample of the current datapoint.

HEATMAP

54.1 Widgets

Higher level widgets that are directly used by the end-user. Both Heatmap widgets use the same plot variant.

54.1.1 HeatmapSplitterWidget

Heatmap with a vertical splitter that can be used to house another widget. The plot is compatible with both ‘row’ and ‘item’ selection modes.

```
class mesmerize.plotting.HeatmapSplitterWidget (highlight_mode='row')
    Widget for interactive heatmaps
```

```
    __init__ (highlight_mode='row')
```

Parameters **highlight_mode** – Interactive mode, one of ‘row’ or ‘item’

```
set_data (dataframes: Union[pandas.core.frame.DataFrame, list], data_column: str, labels_column:
    str, cmap: str = 'jet', transmission: Optional[mesmerize.analysis.data_types.Transmission]
    = None, sort: bool = True, reset_sorting: bool = True, **kwargs)
    Set the data and then set the plot
```

Parameters

- **dataframes** – list of dataframes or a single DataFrame
- **data_column** – data column of the dataframe that is plotted in the heatmap
- **labels_column** – dataframe column (usually categorical labels) used to generate the y-labels and legend.
- **cmap** – colormap choice
- **transmission** – transmission object that dataframe originates, used to calculate data units if passed
- **sort** – if False, the sort comboBox is ignored
- **reset_sorting** – reset the order of the rows in the heatmap
- **kwargs** – Passed to Heatmap.set

```
    _set_sort_order (column: str)
```

Set the sort order of the heatmap rows according to a dataframe column. The column must contain categorical values. The rows are grouped together according to the categorical values.

Parameters **column** – DataFrame column containing categorical values used for sorting the heatmap rows

set_transmission (*transmission: mesmerize.analysis.data_types.Transmission*)

Set the input transmission

get_transmission () → *mesmerize.analysis.data_types.Transmission*

Get the input transmission

highlight_row (*ix: int*)

Highlight a row on the heatmap

54.1.2 HeatmapTracerWidget

Heatmap with an embedded *Datapoint Tracer* that can be saved and restored.

class *mesmerize.plotting.HeatmapTracerWidget*

Bases: *mesmerize.plotting.widgets.base.BasePlotWidget*, *mesmerize.plotting.widgets.heatmap.widget.HeatmapSplitterWidget*

Heatmap with an embedded datapoint tracer

drop_opts = ['dataframes', 'transmission']

keys of the plot_opts dict that are not JSON compatible and not required for restoring this plot

live_datapoint_tracer

The embedded *Datapoint Tracer* <API_DatapointTracer>

set_current_datapoint (*ix: tuple*)

Set the currently selected datapoint in the *Datapoint Tracer*.

Parameters **ix** – index, (x, y). x is always 0 for this widget since it only uses ‘row’ selection mode and not ‘item’

get_plot_opts (*drop: bool = False*) → *dict*

Get the plot options

Parameters **drop** – Drop the non-json compatible objects that are not necessary to restore this plot

update_plot ()

Calls set_data and passes dict from get_plot_opts() as keyword arguments

get_cluster_kwargs () → *dict*

Organize kwargs for visualization of hierarchical clustering

set_data (*args, *datapoint_tracer_curve_column: str = None*, **kwargs)

Set the plot data, parameters and draw the plot. If the input Transmission comes directly from the FCluster it will pass a dict from get_cluster_kwargs() to the cluster_kwargs argument. Else it will pass None to cluster_kwargs.

Parameters

- **args** – arguments to pass to superclass set_data() method
- **datapoint_tracer_curve_column** – Data column containing curves to use in the datapoint tracer
- **kwargs** – keyword arguments, passed to superclass set_data() method

property transmission

The input transmission

Return type *Transmission*

save_plot_dialog (*path*, **args*)

Plot save dialog

save_plot (*path*)

Save the plot as a Transmission in an HDF5 file. Plot parameters are stored as a JSON string within the HDF5 file. See `Transmission.to_hdf5`

Parameters **path** – Path to save the file to. For easy identification use “.ptrn” extension.

open_plot_dialog (*filepath*, *dirpath*, **args*, ***kwargs*)

Open plot dialog

open_plot (*ptrn_path*: *str*, *proj_path*: *str*) → Optional[Tuple[*str*, *str*]]

Open a plot saved by the `save_plot()` method

Parameters

- **ptrn_path** – Path to the HDF5 Transmission file. By convention file extension is “.ptrn”
- **proj_path** – Project path of the associated plot data.

54.2 Variant

Lower level widget that handles the actual plotting and user interaction

class `mesmerize.plotting.variants.Heatmap` (*highlight_mode*='row')

Bases: `mesmerize.pyqtgraphCore.widgets.MatplotlibWidget.MatplotlibWidget`

Heatmap plot variant

sig_selection_changed

Emits indices of data coordinates (x, y) from mouse-click events on the heatmap

__init__ (*highlight_mode*='row')

Parameters **highlight_mode** – The selection mode for the heatmap. One of either ‘row’ or ‘item’

data

2D numpy array of the heatmap data

selector

Selection instance that organizes mouse click events on the heatmap

plot

ClusterGrid object instance containing the plot Axes

set (*data*: *numpy.ndarray*, **args*, *ylabels*: Union[*pandas.core.series.Series*, *numpy.ndarray*, *list*] = None, *ylabels_cmap*: *str* = 'tab20', *cluster_kwargs*: *dict* = None, ***kwargs*)

Parameters

- **data** – 2D numpy array
- **args** – Additional args that are passed to `sns.heatmap()`
- **ylabels** – Labels used to create the ylabels bar
- **ylabels_cmap** – colormap for the ylabels bar
- **cluster_kwargs** – keyword arguments for visualizing hierarchical clustering
- **kwargs** – Additional kwargs that are passed to `sns.heatmap()`

add_stimulus_indicator (*start: int, end: int, color: str*)

Add lines to indicate the start and end of a stimulus or behavioral period

Parameters

- **start** – start index
- **end** – end index
- **color** – line color

KSHAPE

```
class mesmerize.plotting.KShapeWidget (parent=None)
    Bases:      PyQt5.QtWidgets.QMainWindow,      mesmerize.plotting.widgets.base.
               BasePlotWidget

    User-end KShape widget

    sig_output_changed
        Emits output Transmission containing cluster labels

    drop_opts = None
        Unused by this plot widget

    property input_arrays
        The input arrays for clustering

            Returns 2D array, shape is [num_samples, padded_peak_curve_length]

            Return type np.ndarray

    property ks
        tslearn KShape object

    property n_clusters
        Number of clusters

            Return type int

    property train_data
        The training data for clustering

            Returns Training data as a 2D array, shape is [n_samples, padded_curve_length]

            Return type np.ndarray

    property y_pred
        Predicted cluster labels after the model converges

            Returns 1D array of cluster labels that correspond to the input_data

            Return type np.ndarray

    property cluster_centers
        Cluster centroids

            Returns 2D array, shape is [n_clusters, centroid_array]

            Return type np.ndarray

    property cluster_means
        The cluster means
```

Returns 2D array, shape is [cluster_label, mean_array]

Return type np.ndarray

property params

Parameters dict.

Return type dict

set_input (*transmission: mesmerize.analysis.data_types.Transmission*)

Set the input Transmission for the widget

Parameters transmission – Input Transmission

pad_input_data (*a: numpy.ndarray, method: str = 'random'*) → numpy.ndarray

Pad all the input arrays so that are of the same length. The length is determined by the largest input array. The padding value for each input array is the minimum value in that array.

Padding for each input array is either done after the array's last index to fill up to the length of the largest input array (method 'fill-size') or the padding is randomly flanked to the input array (method 'random') for easier visualization.

Parameters

- **a** – 1D array of input arrays where each element is a sample array
- **method** – 'fill-size' or 'random'

Returns 2D array of the padded arrays in the rows

start_process (**args, **kwargs*)

Start the the KShape clustering in a QProcess

property transmission

The input transmission

Return type *Transmission*

class mesmerize.plotting.widgets.kshape.widget.**KShapeMeansPlot** (*parent*)

Bases: mesmerize.pyqtgraphCore.widgets.MatplotlibWidget.MatplotlibWidget

Means plots grouped by cluster membership

axis

array of axis objects used for drawing the means plots, shape is [nrows, ncols]

set_plots (*input_arrays: numpy.ndarray, n_clusters: int, y_pred: numpy.ndarray, xzero_pos: str, error_band*)

Set the subplots

Parameters

- **input_arrays** – padded input arrays (2D), shape is [num_samples, padded_peak_curve_length]
- **n_clusters** – number of clusters
- **y_pred** – cluster predictions (labels)
- **xzero_pos** – set the zero position as the 'zero' position of the input array or the 'maxima' of the input array
- **error_band** – Type of error band to show, one of either 'ci' or 'std'

class mesmerize.plotting.widgets.kshape.widget.**KShapePlot** (*parent*)

Bases: PyQt5.QtWidgets.QDockWidget

Curves plots, showing a sample of individual curves from a single cluster

ax

The Axes object for this plot

plot

MatplotlibWidget() instance

PROPORTIONS

```
class mesmerize.plotting.ProportionsWidget
    Bases:      mesmerize.plotting.widgets.base.BasePlotWidget,      mesmerize.
               pyqtgraphCore.widgets.MatplotlibWidget.MatplotlibWidget

    drop_opts = ['xs', 'ys']
        Drop the 'xs' and 'ys' since they are pd.Series objects and not required for restoring the plot

    property ax
        The Axes object for this plot

        Returns The Axes object for this plot

        Return type AXes

    get_plot_opts (drop: bool = False)
        Get the plot options

        Parameters drop – Drop the 'xs' and 'ys' objects when saving the returned dict for saving
            to an hdf5 file

    update_plot ()
        Update the plot data and draw

    export (*args, **kwargs)
        Export as a csv file

    property transmission
        The input transmission

        Return type Transmission

    save_plot_dialog (path, *args)
        Plot save dialog

    save_plot (path)
        Save the plot as a Transmission in an HDF5 file. Plot parameters are stored as a JSON string within the
        HDF5 file. See Transmission.to_hdf5

        Parameters path – Path to save the file to. For easy identification use “.ptrn” extension.

    open_plot_dialog (filepath, dirpath, *args, **kwargs)
        Open plot dialog

    open_plot (ptrn_path: str, proj_path: str) → Optional[Tuple[str, str]]
        Open a plot saved by the save_plot() method

        Parameters
```

- **ptrn_path** – Path to the HDF5 Transmission file. By convention file extension is “.ptrn”
- **proj_path** – Project path of the associated plot data.

SCATTER PLOT

57.1 ScatterPlotWidget

Higher level widget that is directly used by the end-user. Scatter plot with docked Control Widget, *Datapoint Tracer*, and Console.

class mesmerize.plotting.ScatterPlotWidget

Bases: PyQt5.QtWidgets.QMainWindow, *mesmerize.plotting.widgets.base.BasePlotWidget*

live_datapoint_tracer

Instance of *DatapointTracer*

set_update_live (*b: bool*)

Must be implemented in subclass

set_input (*args, **kws)

Set the input Transmission with data to plot

Parameters transmission – Input transmission

get_plot_opts (*drop: bool = False*) → dict

Get the plot options

Parameters drop – no drop opts are specified for this plot

set_plot_opts (*args, **kws)

Must be implemented in subclass

update_plot ()

Update the plot data and draw

set_current_datapoint (*identifier: uuid.UUID*)

Set the UUID of the current datapoint and update the Datapoint Tracer

property transmission

The input transmission

Return type *Transmission*

save_plot_dialog (*path, *args*)

Plot save dialog

save_plot (*path*)

Save the plot as a Transmission in an HDF5 file. Plot parameters are stored as a JSON string within the HDF5 file. See *Transmission.to_hdf5*

Parameters path – Path to save the file to. For easy identification use “.ptn” extension.

open_plot_dialog (*filepath, dirpath, *args, **kwargs*)
Open plot dialog

open_plot (*ptrn_path: str, proj_path: str*) → Optional[Tuple[str, str]]
Open a plot saved by the save_plot() method

Parameters

- **ptrn_path** – Path to the HDF5 Transmission file. By convention file extension is “.ptrn”
- **proj_path** – Project path of the associated plot data.

57.2 Variant

Lower level widget that interfaces with pqygraph.ScatterPlotItem and has some helper methods.

class mesmerize.plotting.variants.**PgScatterPlot** (*graphics_view: mesmerize.pyqtgraphCore.widgets.GraphicsLayoutWidget.GraphicsLayoutWidget, parent=None*)

Bases: PyQt5.QtCore.QObject

signal_spot_clicked
Emits the UUID of a spot when it is clicked

__init__ (*graphics_view: mesmerize.pyqtgraphCore.widgets.GraphicsLayoutWidget.GraphicsLayoutWidget, parent=None*)

Parameters **graphics_view** – This plot will instantiate within this GraphicsLayoutWidget

add_data (*xs: numpy.ndarray, ys: numpy.ndarray, uuid_series: pandas.core.series.Series, color: Union[str, PyQt5.QtGui.QColor, PyQt5.QtGui.QBrush, List[Union[PyQt5.QtGui.QBrush, PyQt5.QtGui.QColor, str]]], size: int = 10, **kwargs*)
Add data to the plot

Parameters

- **xs** (*np.ndarray*) – array of x values, indices must correspond to the “ys” array
- **ys** (*np.ndarray*) – array of y values, indices must correspond to the “xs” array
- **uuid_series** (*pd.Series*) – series of UUID values. Each SpotItem on the plot is tagged with these UUIDs, therefore the indices must correspond to the “xs” and “ys” arrays.
- **color** (*Union[str, QtGui.QColor, QtGui.QBrush, List[Union[QtGui.QBrush, QtGui.QColor, str]]]*) – Either a single color or list of colors that pqygraph.fn.mkBrush() can accept
- **size** (*int*) – spot size
- **kwargs** – any additional kwargs that are passed to ScatterPlotItem.addPoints()

_clicked (*plot, points*)
Called when the plot is clicked

set_legend (*colors: dict, shapes: dict = None*)
Set the legend.

Parameters

- **colors** (*Dict[str, Union[QtGui.QColor, QtGui.QBrush, str]]*) – dict mapping of labels onto their corresponding colors {'label': QtGui.QColor}
- **shapes** (*Dict[str, str]*) – dict mapping of labels onto their corresponding shapes {'label': <shape as a str>}

clear_legend()
Clear the legend

clear()
Clear the plot

SPACEMAP

```
class mesmerize.plotting.SpaceMapWidget
    Bases:      PyQt5.QtWidgets.QMainWindow,      mesmerize.plotting.widgets.base.
              BasePlotWidget

    sample_df
        sub-dataframe of the current sample

    update_plot (*args, **kwargs)
        Must be implemented in subclass

    property transmission
        The input transmission

        Return type Transmission

    get_plot_opts (drop: bool = False) → dict
        Must be implemented in subclass

    save_plot_dialog (path, *args)
        Plot save dialog

    save_plot (path)
        Save the plot as a Transmission in an HDF5 file. Plot parameters are stored as a JSON string within the
        HDF5 file. See Transmission.to_hdf5

        Parameters path – Path to save the file to. For easy identification use “.ptrn” extension.

    open_plot_dialog (filepath, dirpath, *args, **kwargs)
        Open plot dialog

    open_plot (ptrn_path: str, proj_path: str) → Optional[Tuple[str, str]]
        Open a plot saved by the save_plot() method

        Parameters

        • ptrn_path – Path to the HDF5 Transmission file. By convention file extension
          is “.ptrn”

        • proj_path – Project path of the associated plot data.

    set_plot_opts (*args, **kws)
        Must be implemented in subclass
```


INDICES AND TABLES

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

PYTHON MODULE INDEX

m

- `mesmerize.analysis.clustering_metrics,`
244
- `mesmerize.analysis.math.cross_correlation,`
241
- `mesmerize.analysis.utils,` 239
- `mesmerize.common,` 195
- `mesmerize.common.qdialogs,` 197
- `mesmerize.common.utils,` 195
- `mesmerize.pyqtgraphCore.flowchart.library.Biology,`
251
- `mesmerize.pyqtgraphCore.flowchart.library.Clustering,`
252
- `mesmerize.pyqtgraphCore.flowchart.library.Data,`
247
- `mesmerize.pyqtgraphCore.flowchart.library.Display,`
248
- `mesmerize.pyqtgraphCore.flowchart.library.Hierarchical,`
252
- `mesmerize.pyqtgraphCore.flowchart.library.Math,`
250
- `mesmerize.pyqtgraphCore.flowchart.library.Signal,`
249
- `mesmerize.pyqtgraphCore.flowchart.library.Transform,`
252

Symbols

<code>__del__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 207
<code>__delitem__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208
<code>__getitem__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCN method), 211
<code>__init__()</code>	(mesmerize.Transmission method), 231	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCN method), 209
<code>__init__()</code>	(mesmerize.analysis.cross_correlation.CC_Data method), 243	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 209
<code>__init__()</code>	(mesmerize.analysis.data_types.BaseTransmission method), 233	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 211
<code>__init__()</code>	(mesmerize.analysis.data_types.HistoryTrace method), 235	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 210
<code>__init__()</code>	(mesmerize.plotting.HeatmapSplitterWidget method), 261	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 212
<code>__init__()</code>	(mesmerize.plotting.variants.Heatmap method), 263	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216
<code>__init__()</code>	(mesmerize.plotting.variants.PgScatterPlot method), 272	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 226
<code>__init__()</code>	(mesmerize.plotting.widgets.base.BasePlotWidget method), 257	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218
<code>__init__()</code>	(mesmerize.viewer.core.ViewerUtils method), 201	<code>__init__()</code>	(mesmerize.viewer.core.data_types.ImgData method), 201
<code>__init__()</code>	(mesmerize.viewer.core.ViewerWorkEnv method), 199	<code>__init__()</code>	(mesmerize.viewer.core.mesfile.MES method), 202
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types), 214	<code>__init__()</code>	(mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203
<code>__init__()</code>	(mesmerize.viewer.core.ViewerWorkEnv attribute), 199	<code>__init__()</code>	(mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCN method), 211
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCN method), 211	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 209
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 209	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 211
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 211	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 210
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScale method), 210	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 212
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 226
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 226	<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218
<code>__init__()</code>	(mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218	<code>__init__()</code>	(mesmerize.viewer.core.data_types.ImgData method), 201
<code>__init__()</code>	(mesmerize.viewer.core.ViewerUtils method), 201	<code>__init__()</code>	(mesmerize.viewer.core.mesfile.MES method), 202
<code>__init__()</code>	(mesmerize.viewer.core.ViewerWorkEnv method), 199	<code>__init__()</code>	(mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203

ize.viewer.modules.roi_manager_modules.roi_types.ScatterROI, *graphics_object()* (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 220
 __init__() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 213
 ize.viewer.modules.roi_manager_modules.roi_types.VolCNMF() (mesmerize.analysis.data_types.HistoryTrace static method), 236
 __init__() (mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI method), 214
 __weakref__ (mesmerize.analysis.data_types.HistoryTrace attribute), 237
 __weakref__ (mesmerize.viewer.core.ViewerUtils attribute), 201
 __weakref__ (mesmerize.viewer.core.ViewerWorkEnv attribute), 199
 __weakref__ (mesmerize.viewer.core.mesfile.MES attribute), 202
 __weakref__ (mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager attribute), 208
 __weakref__ (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList attribute), 214
 __weakref__ (mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI attribute), 215
 _add_data_block() (mesmerize.analysis.data_types.HistoryTrace method), 236
 _clear_workEnv() (mesmerize.viewer.core.ViewerUtils method), 201
 _clicked() (mesmerize.plotting.variants.PgScatterPlot method), 272
 _hide_all_graphics_objects() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 _hide_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 _load_files() (mesmerize.Transmission static method), 232
 _organize_meta() (mesmerize.viewer.core.ViewerWorkEnv static method), 199
 _reindex_list_widget() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 _set_sort_order() (mesmerize.plotting.HeatmapSplitterWidget method), 261
 _show_all_graphics_objects() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 AbsoluteValue (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250
 AbstractBaseManager (class in mesmerize.viewer.modules.roi_manager_modules.managers), 208
 add_all_components() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNMF method), 212
 add_all_components() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMF method), 211
 add_data() (mesmerize.plotting.variants.PgScatterPlot method), 272
 add_item() (mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203
 add_manual_roi() (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
 add_operation() (mesmerize.analysis.data_types.HistoryTrace method), 236
 add_roi() (mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208
 add_roi() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNMF method), 212
 add_roi() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMF method), 209
 add_roi() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScatterROI method), 210
 add_roi() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMF method), 211
 add_row() (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229
 add_stimulus_indicator() (mesmerize.plotting.variants.Heatmap method), 263
 add_to_batch() (mesmerize.viewer.modules.cnmf.ModuleGUI method),

205
 add_to_batch_cnmfe() (mesmerize.viewer.modules.cnmfe.ModuleGUI method), 206
 add_to_batch_corr_pnr() (mesmerize.viewer.modules.cnmfe.ModuleGUI method), 206
 add_to_batch_elas_corr() (mesmerize.viewer.modules.caiman_motion_correction.ModuleGUI method), 205
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI method), 215
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 217
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMROI method), 227
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 222
 add_to_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VoIGMF method), 224
 AnalysisGraph (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
 append() (mesmerize.viewer.modules.roi_manager_modules.roi_list_manager.ROIList method), 213
 ArgGroupStat (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250
 ArrayStats (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250
 auto_colormap() (in module mesmerize.plotting.utils), 255
 ax (mesmerize.plotting.widgets.kshape.widget.KShapePlot attribute), 267
 ax() (mesmerize.plotting.ProportionsWidget property), 269
 axs (mesmerize.plotting.widgets.kshape.widget.KShapeMeansPlot attribute), 266

B

BasePlotWidget (class in mesmerize.plotting.widgets.base), 257
 BaseROI (class in mesmerize.viewer.modules.roi_manager_modules.roi_types), 216

BaseTransmission (class in mesmerize.analysis.data_types), 233
 BeeswarmPlots (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
 block_signals_list (mesmerize.plotting.widgets.base.BasePlotWidget attribute), 258
 BlockGUIWorth (class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 249
 CC_Data (class in mesmerize.analysis.cross_correlation), 243
 ccs (mesmerize.analysis.cross_correlation.CC_Data attribute), 243
 CNMROI_path() (mesmerize.viewer.modules.tiff_io.ModuleGUI method), 204
 cluster_centers() (mesmerize.analysis.data_types.HistoryTrace method), 236
 cluster_means() (mesmerize.analysis.data_types.HistoryTrace static method), 237
 clear() (mesmerize.viewer.core.ViewerWorkEnv method), 199
 clear() (mesmerize.viewer.modules.roi_manager_modules.managers.AbstractionManager.ROIList method), 208
 clear() (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229
 clear_() (mesmerize.viewer.modules.roi_manager_modules.roi_list_manager.ROIList method), 213
 clear_legend() (mesmerize.plotting.variants.PgScatterPlot method), 273
 cluster_centers() (mesmerize.plotting.KShapeWidget property), 265
 cluster_means() (mesmerize.plotting.KShapeWidget property), 265
 CNMROI (class in mesmerize.viewer.modules.roi_manager_modules.roi_types), 226
 compute_cc_data() (in module mesmerize.analysis.math.cross_correlation), 242
 compute_ccs() (in module mesmerize.analysis.math.cross_correlation), 242
 create_data_block() (mesmerize.analysis.data_types.HistoryTrace method), 236
 create_roi_list() (mesmer-

`ize.viewer.modules.roi_manager_modules.managers.ManageROI()` (mesmerize.viewer.core.ViewerUtils method), 201
`create_roi_list()` (mesmerize.viewer.modules.roi_manager_modules.managers.ManageROI method), 209
`create_roi_list()` (mesmerize.viewer.modules.roi_manager_modules.managers.ManageROI method), 210
`create_roi_list()` (mesmerize.viewer.modules.roi_manager_modules.managers.ManageROI method), 211
`create_roi_list()` (mesmerize.viewer.modules.roi_manager_modules.managers.ManageROI method), 210
`CrossCorr` (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
`CtrlNode` (class in mesmerize.pyqtgraphCore.flowchart.library.common), 253
`ctrlWidget()` (mesmerize.pyqtgraphCore.flowchart.library.common.CtrlNode method), 253
`current_index` (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList attribute), 212
`curve_data()` (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI property), 218
`curve_uuids` (mesmerize.analysis.cross_correlation.CC_Data attribute), 243
D
`data` (mesmerize.plotting.variants.Heatmap attribute), 263
`data_blocks()` (mesmerize.analysis.data_types.HistoryTrace property), 236
`DatapointTracerWidget` (class in mesmerize.plotting), 259
`davies_bouldin_score()` (in module mesmerize.analysis.clustering_metrics), 245
`del_item()` (mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203
`delete_row()` (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229
`Derivative` (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250
`df` (mesmerize.viewer.modules.batch_manager.ModuleGUI attribute), 203
`disconnect_all()` (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
`draw_graph()` (in module mesmerize.common.utils), 185
`draw_graph()` (mesmerize.analysis.data_types.HistoryTrace method), 265
`drop_opts` (mesmerize.plotting.HeatmapTracerWidget attribute), 262
`drop_opts` (mesmerize.plotting.KShapeWidget attribute), 265
`drop_opts` (mesmerize.plotting.ProportionsWidget attribute), 269
`DropNa` (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 247
E
`epsilon_matrix` (mesmerize.analysis.cross_correlation.CC_Data attribute), 243
`eventFilter()` (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
`exceptions_label()` (in module mesmerize.common.qdialogs), 197
`export()` (mesmerize.plotting.ProportionsWidget method), 269
`ExtractStim` (class in mesmerize.pyqtgraphCore.flowchart.library.Biology), 251
F
`FCluster` (class in mesmerize.pyqtgraphCore.flowchart.library.Hierarchical), 252
`fill_control_widget()` (mesmerize.plotting.widgets.base.BasePlotWidget method), 258
`FrequencyDomainMagnitude` (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
`from_dict()` (mesmerize.analysis.cross_correlation.CC_Data class method), 244

`from_dict()` (*mesmerize.analysis.data_types.HistoryTrace* static method), 237
`from_hdf5()` (*mesmerize.analysis.cross_correlation.CC_Data* class method), 244
`from_hdf5()` (*mesmerize.analysis.data_types.BaseTransmission* class method), 234
`from_hdf5()` (*mesmerize.Transmission* class method), 232
`from_json()` (*mesmerize.analysis.data_types.HistoryTrace* class method), 237
`from_mesfile()` (*mesmerize.viewer.core.ViewerWorkEnv* class method), 199
`from_pickle()` (*mesmerize.analysis.data_types.BaseTransmission* class method), 234
`from_pickle()` (*mesmerize.analysis.data_types.HistoryTrace* class method), 237
`from_pickle()` (*mesmerize.Transmission* class method), 232
`from_pickle()` (*mesmerize.viewer.core.ViewerWorkEnv* class method), 200
`from_proj()` (*mesmerize.Transmission* class method), 232
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI* class method), 215
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI* class method), 217
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI* class method), 227
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI* class method), 218
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI* class method), 221
`from_state()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMFROI* class method), 225
`from_tiff()` (*mesmerize.viewer.core.ViewerWorkEnv* class method), 200

G
`get_all_data_blocks_history()` (*mesmerize.analysis.data_types.HistoryTrace* method), 236
`get_all_states()` (*mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager* method), 208
`get_all_states()` (*mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNC* method), 212
`get_all_states()` (*mesmerize.viewer.modules.roi_manager_modules.managers.ManagerMan* method), 209
`get_all_states()` (*mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVol* method), 211
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBas* method), 215
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI* method), 217
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI* method), 227
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI* method), 219
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI* method), 222
`get_all_tags()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF* method), 224
`get_centerlike()` (*in module mesmerize.analysis.util*s), 239
`get_centerlike()` (*in module mesmerize.analysis.clustering_metrics*), 244
`get_cluster_kwargs()` (*mesmerize.plotting.HeatmapTracerWidget* method), 216
`get_cluster_radius()` (*in module mesmerize.analysis.clustering_metrics*), 244
`get_color()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBas* method), 214
`get_color()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI* method), 216
`get_color()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI* method), 227
`get_color()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI* method), 219
`get_color()` (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI* method), 222

method), 221

get_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF.plotting.ScatterPlotWidget method), 271

get_colormap() (in module mesmerize.plotting.utils), 255

get_data_block_history() (mesmerize.analysis.data_types.HistoryTrace method), 236

get_dataframe() (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229

get_epsilon() (in module mesmerize.analysis.math.cross_correlation), 241

get_epsilon_matrix() (in module mesmerize.analysis.math.cross_correlation), 242

get_frequency_linspace() (in module mesmerize.analysis.utils), 239

get_image_references() (mesmerize.viewer.core.mesfile.MES method), 202

get_item_index() (mesmerize.viewer.modules.batch_manager.ModuleGUI method), 203

get_lag() (in module mesmerize.analysis.math.cross_correlation), 241

get_lag_matrix() (in module mesmerize.analysis.math.cross_correlation), 242

get_omega() (in module mesmerize.analysis.math.cross_correlation), 241

get_operation_params() (mesmerize.analysis.data_types.HistoryTrace method), 236

get_operations_list() (mesmerize.analysis.data_types.HistoryTrace method), 236

get_params() (mesmerize.viewer.modules.caiman_motion_correction.ModuleGUI method), 205

get_params() (mesmerize.viewer.modules.cnmf.ModuleGUI method), 205

get_params() (mesmerize.viewer.modules.cnmfe.ModuleGUI method), 205

get_plot_item() (mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBaseManager method), 208

get_plot_opts() (mesmerize.plotting.HeatmapTracerWidget method), 262

get_plot_opts() (mesmerize.plotting.ProportionsWidget method), 269

get_plot_opts() (mesmerize.plotting.ScatterPlotWidget method), 271

get_plot_opts() (mesmerize.plotting.SpaceMapWidget method), 275

get_plot_opts() (mesmerize.plotting.widgets.base._AbstractBasePlotWidget method), 257

get_plot_opts() (mesmerize.plotting.widgets.base.BasePlotWidget method), 258

get_proj_config() (in module mesmerize.common), 195

get_proj_path() (mesmerize.analysis.data_types.BaseTransmission method), 234

get_proj_path() (mesmerize.Transmission method), 232

get_project_manager() (in module mesmerize.common), 195

get_proportions() (in module mesmerize.analysis.utils), 239

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 214

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 226

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221

get_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224

get_sampling_rate() (in module mesmerize.analysis.utils), 240

get_state() (mesmerize.plotting.utils.WidgetRegistry method), 256

get_sys_config() (in module mesmerize.common), 195

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 215

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 217

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 227

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221

get_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224

`ize.viewer.modules.roi_manager_modules.roi_types.ManualROI` (mesmerize.viewer.core.ViewerWorkEnv attribute), 200
`get_tag()` (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROIviewer.modules.roi_manager.ModuleGUI method), 207
`get_tag()` (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMFviewer.modules.roi_manager_modules.managers.ManagerMan method), 209
`get_threshold_matrix()` (mesmerize.analysis.cross_correlation.CC_Data Inconsistent (class in mesmerize.pyqtgraphCore.flowchart.library.Hierarchical), 252 method), 243
`get_transmission()` (mesmerize.plotting.HeatmapSplitterWidget method), 265
`get_units()` (mesmerize.viewer.modules.stimmap_modules.page.Page Integrate (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250 method), 229
`get_window_manager()` (in module mesmerize.common), 195
`iRFFT` (class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 250
`is_empty()` (mesmerize.viewer.modules.roi_manager_modules.managers.AbstractBase method), 208
`isEmpty` (mesmerize.viewer.core.ViewerWorkEnv attribute), 200
H
`HdfTools` (class in mesmerize.common.utils), 196
`Heatmap` (class in mesmerize.plotting.variants), 263
`Heatmap` (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
`HeatmapSplitterWidget` (class in mesmerize.plotting), 261
`HeatmapTracerWidget` (class in mesmerize.plotting), 262
`highlight_curve()` (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
`highlight_roi()` (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
`highlight_row()` (mesmerize.plotting.HeatmapSplitterWidget method), 262
`history()` (mesmerize.analysis.data_types.HistoryTrace property), 236
`history_trace` (mesmerize.viewer.core.ViewerWorkEnv attribute), 200
`HistoryTrace` (class in mesmerize.analysis.data_types), 234
I
`iloc` (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 248
`ImgData` (class in mesmerize.viewer.core.data_types), 201
K
`KMeans` (class in mesmerize.pyqtgraphCore.flowchart.library.Clustering), 252
`ks()` (mesmerize.plotting.KShapeWidget property), 265
`KShape` (class in mesmerize.pyqtgraphCore.flowchart.library.Clustering), 252
`KShapeMeansPlot` (class in mesmerize.plotting.widgets.kshape.widget), 266
`KShapePlot` (class in mesmerize.plotting.widgets.kshape.widget), 266
`KShapeWidget` (class in mesmerize.plotting), 265
L
`labels` (mesmerize.analysis.cross_correlation.CC_Data attribute), 243
`lag_matrix` (mesmerize.analysis.cross_correlation.CC_Data attribute), 243
`LDA` (class in mesmerize.pyqtgraphCore.flowchart.library.Transform), 252
`Linkage` (class in mesmerize.pyqtgraphCore.flowchart.library.Hierarchical), 252
`LinRegress` (class in mesmerize.pyqtgraphCore.flowchart.library.Math), 250

list_widget (mesmerize.viewer.modules.roi_manager_modules.managers),
ize.viewer.modules.roi_manager_modules.roi_list.ROIList 209
attribute), 212

list_widget_tags (mesmerize.viewer.modules.roi_manager_modules.managers),
ize.viewer.modules.roi_manager_modules.roi_list.ROIList 211
attribute), 212

live_datapoint_tracer (mesmerize.viewer.modules.roi_manager_modules.managers),
ize.plotting.HeatmapTracerWidget attribute), 210

live_datapoint_tracer (mesmerize.pyqtgraphCore.flowchart.library.Transform),
ize.plotting.ScatterPlotWidget attribute), 252

load() (mesmerize.viewer.modules.tiff_io.ModuleGUI
method), 204

load_dataframe() (mesmerize.common.utils.HdfTools static
method), 196

load_dict() (mesmerize.common.utils.HdfTools
static method), 196

load_img() (mesmerize.viewer.core.mesfile.MES
method), 202

load_item_input() (mesmerize.viewer.modules.batch_manager.ModuleGUI
method), 203

load_item_output() (mesmerize.viewer.modules.batch_manager.ModuleGUI
method), 204

load_mesfile() (mesmerize.viewer.core.ViewerWorkEnv static
method), 200

LoadFile (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
247

LoadProjDF (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
247

LogTransform (class in mesmerize.pyqtgraphCore.flowchart.library.Math),
251

M

make_runfile() (in module mesmerize.common.utils), 195

make_workdir() (in module mesmerize.common.utils), 195

manager (mesmerize.viewer.modules.roi_manager.ModuleGUI
attribute), 207

ManagerCNMFROI (class in mesmerize.viewer.modules.roi_manager_modules.managers),
211

ManagerManual (class in mesmerize.viewer.modules.roi_manager_modules.managers),
209

ManagerScatterROI (class in mesmerize.viewer.modules.roi_manager_modules.managers),
211

ManagerVolCNMF (class in mesmerize.viewer.modules.roi_manager_modules.managers),
210

ManagerVolROI (class in mesmerize.pyqtgraphCore.flowchart.library.Transform),
252

ManualDFoF (class in mesmerize.pyqtgraphCore.flowchart.library.Biology),
251

ManualROI (class in mesmerize.viewer.modules.roi_manager_modules.roi_types),
218

map_labels_to_colors() (in module mesmerize.plotting.utils), 255

maps() (mesmerize.viewer.modules.stimulus_mapping.ModuleGUI
property), 229

MaxInconsistent (class in mesmerize.pyqtgraphCore.flowchart.library.Hierarchical),
252

MaxIncStat (class in mesmerize.pyqtgraphCore.flowchart.library.Hierarchical),
252

Merge (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
247

merge() (mesmerize.analysis.data_types.HistoryTrace
class method), 237

merge() (mesmerize.Transmission class method), 232

MES (class in mesmerize.viewer.core.mesfile), 202

mesmerize.analysis.clustering_metrics
module, 244

mesmerize.analysis.math.cross_correlation
module, 241

mesmerize.analysis.utils
module, 239

mesmerize.common
module, 195

mesmerize.common.qdialogs
module, 197

mesmerize.common.utils
module, 195

mesmerize.pyqtgraphCore.flowchart.library.Biology
module, 251

mesmerize.pyqtgraphCore.flowchart.library.Clusterin
module, 252

mesmerize.pyqtgraphCore.flowchart.library.Data
module, 247

mesmerize.pyqtgraphCore.flowchart.library.Display
module, 248

mesmerize.pyqtgraphCore.flowchart.library.NoHierarchical (class in mesmerize.pyqtgraphCore.flowchart.library.Signal),
 module, 252
 mesmerize.pyqtgraphCore.flowchart.library.Math (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
 module, 250
 NormRaw (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
 module, 249
 247
 mesmerize.pyqtgraphCore.flowchart.library.Transform (class in mesmerize.pyqtgraphCore.flowchart.library.Data),
 module, 252
O
 module
 open_in_viewer () (mesmerize.plotting.DatapointTracerWidget method),
 mesmerize.analysis.clustering_metrics, 244
 259
 mesmerize.analysis.math.cross_correlation, 241
 open_plot () (mesmerize.plotting.HeatmapTracerWidget method),
 263
 mesmerize.analysis.utils, 239
 mesmerize.common, 195
 open_plot () (mesmerize.plotting.ProportionsWidget method), 269
 mesmerize.common.qdialogs, 197
 open_plot () (mesmerize.plotting.ScatterPlotWidget method), 272
 mesmerize.common.utils, 195
 mesmerize.pyqtgraphCore.flowchart.library.Biology, 251
 open_plot () (mesmerize.plotting.SpaceMapWidget method), 275
 mesmerize.pyqtgraphCore.flowchart.library.Chemistry, 252
 open_plot () (mesmerize.plotting.widgets.base._AbstractBasePlotWidget method), 257
 mesmerize.pyqtgraphCore.flowchart.library.Date, 247
 many.Display, (mesmerize.plotting.widgets.base.BasePlotWidget method), 258
 mesmerize.pyqtgraphCore.flowchart.library.Histology, 252
 open_plot_dialog () (mesmerize.plotting.HeatmapTracerWidget method), 263
 mesmerize.pyqtgraphCore.flowchart.library.Math, 250
 many.SignalDialog () (mesmerize.plotting.ProportionsWidget method), 269
 mesmerize.pyqtgraphCore.flowchart.library.Transform, 252
 open_plot_dialog () (mesmerize.plotting.ScatterPlotWidget method), 271
 ModuleGUI (class in mesmerize.viewer.modules.batch_manager), 203
 open_plot_dialog () (mesmerize.plotting.SpaceMapWidget method), 275
 ModuleGUI (class in mesmerize.viewer.modules.caiman_motion_correction), 205
 open_plot_dialog () (mesmerize.plotting.widgets.base.BasePlotWidget method), 258
 ModuleGUI (class in mesmerize.viewer.modules.cnmf), 205
 organize_dataframe_columns () (in module mesmerize.analysis.utils), 240
 ModuleGUI (class in mesmerize.viewer.modules.cnmf), 205
P
 ModuleGUI (class in mesmerize.viewer.modules.roi_manager), 207
 package_for_project () (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
 ModuleGUI (class in mesmerize.viewer.modules.stimulus_mapping), 229
 pad_arrays () (in module mesmerize.analysis.utils), 240
 ModuleGUI (class in mesmerize.viewer.modules.tiff_io), 204
 pad_input_data () (mesmerize.plotting.KShapeWidget method), 266
 n_clusters () (mesmerize.plotting.KShapeWidget property), 265
 PadArrays (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 247
 ncc_c () (in module mesmerize.analysis.math.cross_correlation), 241

Page	(class in mesmerize.viewer.modules.stimmap_modules.page), 229	method), 213
params()	(mesmerize.plotting.KShapeWidget property), 266	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 215
PCA	(class in mesmerize.pyqtgraphCore.flowchart.library.Transform), 252	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 217
PeakDetect	(class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 249	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 227
PeakFeatures	(class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 249	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
PgScatterPlot	(class in mesmerize.plotting.variants), 272	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 222
Plot	(class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248	remove_from_viewer() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 225
plot	(mesmerize.plotting.variants.Heatmap attribute), 263	Resample (class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 249
plot	(mesmerize.plotting.widgets.kshape.widget.KShapePlot attribute), 267	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 214
plot_manual_roi_regions()	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216
plot_state	(mesmerize.analysis.data_types.BaseTransmission attribute), 233	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 227
PowerSpectralDensity	(class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 249	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
present_exceptions()	(in module mesmerize.common.qdialogs), 197	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221
previous_index	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList attribute), 213	reset_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224
process()	(mesmerize.pyqtgraphCore.flowchart.library.common.CtrlNode method), 253	restore_from_states() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNMF method), 212
process_batch()	(mesmerize.viewer.modules.batch_manager.ModuleGUI method), 204	restore_from_states() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerManualROI method), 209
Proportions	(class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248	restore_from_states() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScatterROI method), 210
ProportionsWidget	(class in mesmerize.plotting), 269	restore_from_states() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMF method), 211
R		restoreState() (mesmerize.pyqtgraphCore.flowchart.library.common.CtrlNode method), 253
register()	(mesmerize.plotting.utils.WidgetRegistry method), 256	
reindex_colormap()	(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213	

method), 253

RFFT (class in *mesmerize.pyqtgraphCore.flowchart.library.Signal*), 249

roi_list (*mesmerize.viewer.modules.roi_manager_modules.manager* attribute), 208

roi_manager (*mesmerize.viewer.core.ViewerWorkEnv* attribute), 200

ROIList (class in *mesmerize.viewer.modules.roi_manager_modules.roi_list*), 212

S

sample_df (*mesmerize.plotting.SpaceMapWidget* attribute), 275

sample_id (*mesmerize.viewer.core.ViewerWorkEnv* attribute), 200

Save (class in *mesmerize.pyqtgraphCore.flowchart.library.Data*), 247

save_dataframe() (*mesmerize.common.utils.HdfTools* static method), 196

save_dict() (*mesmerize.common.utils.HdfTools* static method), 196

save_plot() (*mesmerize.plotting.HeatmapTracerWidget* method), 263

save_plot() (*mesmerize.plotting.ProportionsWidget* method), 269

save_plot() (*mesmerize.plotting.ScatterPlotWidget* method), 271

save_plot() (*mesmerize.plotting.SpaceMapWidget* method), 275

save_plot() (*mesmerize.plotting.widgets.base._AbstractBasePlotWidget* method), 257

save_plot() (*mesmerize.plotting.widgets.base.BasePlotWidget* method), 258

save_plot_dialog() (*mesmerize.plotting.HeatmapTracerWidget* method), 262

save_plot_dialog() (*mesmerize.plotting.ProportionsWidget* method), 269

save_plot_dialog() (*mesmerize.plotting.ScatterPlotWidget* method), 271

save_plot_dialog() (*mesmerize.plotting.SpaceMapWidget* method), 275

save_plot_dialog() (*mesmerize.plotting.widgets.base.BasePlotWidget* method), 258

saveState() (*mesmerize.pyqtgraphCore.flowchart.library.common.CtrlNode* method), 253

SavitzkyGolay (class in *mesmerize.pyqtgraphCore.flowchart.library.Signal*), 249

AbstractBaseManager (class in *mesmerize.pyqtgraphCore.flowchart.library.Signal*), 249

ScalerMeanVariance (class in *mesmerize.pyqtgraphCore.flowchart.library.Signal*), 249

ScatterPlot (class in *mesmerize.pyqtgraphCore.flowchart.library.Display*), 248

ScatterPlotWidget (class in *mesmerize.plotting*), 271

ScatterROI (class in *mesmerize.viewer.modules.roi_manager_modules.roi_types*), 220

SelectColumns (class in *mesmerize.pyqtgraphCore.flowchart.library.Data*), 247

selector (*mesmerize.plotting.variants.Heatmap* attribute), 263

SelectRows (class in *mesmerize.pyqtgraphCore.flowchart.library.Data*), 247

set() (*mesmerize.plotting.variants.Heatmap* method), 263

set_all_from_states() (*mesmerize.viewer.modules.roi_manager.ModuleGUI* method), 207

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI* method), 215

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI* method), 216

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI* method), 227

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI* method), 219

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI* method), 221

set_color() (*mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF* method), 224

set_current_datapoint() (*mesmerize.plotting.HeatmapTracerWidget* method), 262

set_current_datapoint() (*mesmerize.plotting.ScatterPlotWidget* method), 271

set_current_index() (*mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList*

method), 213
 set_curve_data() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 224
 set_curve_data() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 228
 set_curve_data() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 221
 set_curve_data() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 225
 set_data() (mesmerize.plotting.HeatmapSplitterWidget method), 261
 set_data() (mesmerize.plotting.HeatmapTracerWidget method), 262
 set_data() (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229
 set_image() (mesmerize.plotting.DatapointTracerWidget method), 259
 set_input() (mesmerize.plotting.KShapeWidget method), 266
 set_input() (mesmerize.plotting.ScatterPlotWidget method), 271
 set_input() (mesmerize.plotting.widgets.base._AbstractBasePlotWidget method), 257
 set_input() (mesmerize.plotting.widgets.base.BasePlotWidget method), 258
 set_legend() (mesmerize.plotting.variants.PgScatterPlot method), 272
 set_list_widget_tags() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 214
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 227
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221
 set_original_color() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMFROI method), 224
 set_plot_opts() (mesmerize.plotting.SpaceMapWidget method), 275
 set_plot_opts() (mesmerize.plotting.widgets.base._AbstractBasePlotWidget method), 257
 set_plot_opts() (mesmerize.plotting.widgets.base.BasePlotWidget method), 258
 set_plots() (mesmerize.plotting.widgets.kshape.widget.KShapeMeansPlot method), 266
 set_previous_index() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 set_proj_config() (mesmerize.analysis.data_types.BaseTransmission method), 234
 set_proj_path() (mesmerize.analysis.data_types.BaseTransmission method), 234
 set_proj_path() (mesmerize.Transmission method), 232
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types._AbstractBaseROI method), 214
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.BaseROI method), 216
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFROI method), 227
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 218
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221
 set_roi_graphics_object() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMFROI method), 224
 set_spot_size() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerScatterROI method), 210
 set_spot_size() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolCNMFROI method), 211
 set_state() (mesmerize.plotting.utils.WidgetRegistry

method), 256
 set_statusbar() (mesmerize.viewer.core.ViewerUtils method), 202
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI method), 215
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.SpliceROI method), 217
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.GNMROI method), 227
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221
 set_tag() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI method), 215
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.SpliceROI method), 217
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.GNMROI method), 227
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI method), 219
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI method), 221
 set_text() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224
 set_transmission() (mesmerize.plotting.HeatmapSplitterWidget method), 261
 set_units() (mesmerize.viewer.modules.stimmap_modules.page.Page method), 229
 set_update_live() (mesmerize.plotting.ScatterPlotWidget method), 271
 set_widget() (mesmerize.plotting.DatapointTracerWidget method), 259
 set_zlevel() (mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVolROI method), 210
 set_zlevel() (mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMF method), 224
 show_item_info() (mesmerize.viewer.modules.batch_manager.ModuleGUI attribute), 204
 sig_output_changed (mesmerize.plotting.KShapeWidget attribute), 265
 slice_selection_changed (mesmerize.plotting.variants.Heatmap attribute), 263
 Signal (class in mesmerize.pyqtgraphCore.flowchart.library.Signal), 250
 Slocker() (mesmerize.plotting.widgets.base.BasePlotWidget class method), 258
 ScatterROI.clicked (mesmerize.plotting.variants.PgScatterPlot attribute), 272
 VolCNMF.delete_roi_menu() (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
 VolCNMF.checkbox_clicked() (mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList method), 213
 SpliceROI (class in mesmerize.pyqtgraphCore.flowchart.library.Display), 248
 SpliceROIWidget (class in mesmerize.plotting), 275
 SpliceArrays (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 251
 start_backend() (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
 start_manual_mode() (mesmerize.viewer.modules.roi_manager.ModuleGUI method), 207
 start_process() (mesmerize.plotting.KShapeWidget method), 266
 StaticDFoFo (class in mesmerize.pyqtgraphCore.flowchart.library.Biology), 251
 stim_maps (mesmerize.viewer.core.ViewerWorkEnv attribute), 200
 StimTuning (class in mesmerize.pyqtgraphCore.flowchart.library.Biology), 251
 TextFilter (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 247
 Timeseries (class in mesmerize.pyqtgraphCore.flowchart.library.Data), 247

T

[ize.pyqtgraphCore.flowchart.library.Display](#)),
[248](#)
[to_dict\(\)](#) ([mesmerize.analysis.data_types.BaseTransmission](#)
[method](#)), [234](#)
[to_dict\(\)](#) ([mesmerize.analysis.data_types.HistoryTrace](#) [method](#)),
[236](#)
[to_dict\(\)](#) ([mesmerize.Transmission](#) [method](#)), [232](#)
[to_hdf5\(\)](#) ([mesmerize.analysis.cross_correlation.CC_Data](#)
[method](#)), [244](#)
[to_hdf5\(\)](#) ([mesmerize.analysis.data_types.BaseTransmission](#)
[method](#)), [234](#)
[to_hdf5\(\)](#) ([mesmerize.Transmission](#) [method](#)), [232](#)
[to_json\(\)](#) ([mesmerize.analysis.data_types.HistoryTrace](#) [method](#)),
[237](#)
[to_pandas\(\)](#) ([mesmerize.viewer.core.ViewerWorkEnv](#)
[method](#)), [200](#)
[to_pickle\(\)](#) ([mesmerize.analysis.data_types.BaseTransmission](#)
[method](#)), [234](#)
[to_pickle\(\)](#) ([mesmerize.analysis.data_types.HistoryTrace](#) [method](#)),
[237](#)
[to_pickle\(\)](#) ([mesmerize.Transmission](#) [method](#)), [232](#)
[to_pickle\(\)](#) ([mesmerize.viewer.core.ViewerWorkEnv](#)
[method](#)), [201](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.AbstractBaseROI](#)
[method](#)), [215](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.BackROI](#)
[method](#)), [217](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.CNMFRQROI](#)
[method](#)), [228](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.ManualROI](#)
[method](#)), [218](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.ScatterROI](#)
[method](#)), [221](#)
[to_state\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.roi_types.VolCNMFRQROI](#)
[method](#)), [224](#)
[train_data\(\)](#) ([mesmerize.plotting.KShapeWidget](#)
[property](#)), [265](#)
[Transmission](#) (class in [mesmerize](#)), [231](#)
[transmission\(\)](#) ([mesmerize.plotting.HeatmapTracerWidget](#) [property](#)),
[262](#)
[transmission\(\)](#) ([mesmerize.plotting.KShapeWidget](#)
[property](#)), [266](#)
[transmission\(\)](#) ([mesmerize.plotting.ProportionsWidget](#) [property](#)),
[269](#)
[transmission\(\)](#) ([mesmerize.plotting.ScatterPlotWidget](#) [property](#)),
[271](#)
[transmission\(\)](#) ([mesmerize.plotting.SpaceMapWidget](#) [property](#)),
[275](#)
[transmission\(\)](#) ([mesmerize.plotting.widgets.base._AbstractBasePlotWidget](#)
[property](#)), [257](#)
[transmission\(\)](#) ([mesmerize.plotting.widgets.base.BasePlotWidget](#)
[property](#)), [258](#)
[TVDiff](#) (class in [mesmerize.pyqtgraphCore.flowchart.library.Math](#)),
[251](#)

U

[update_idx_components\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.managers.ManagerCNI](#)
[method](#)), [212](#)
[update_idx_components\(\)](#) ([mesmerize.viewer.modules.roi_manager_modules.managers.ManagerVol](#)
[method](#)), [211](#)
[update_plot\(\)](#) ([mesmerize.plotting.HeatmapTracerWidget](#) [method](#)),
[262](#)
[update_plot\(\)](#) ([mesmerize.plotting.ProportionsWidget](#) [method](#)),
[269](#)
[update_plot\(\)](#) ([mesmerize.plotting.ScatterPlotWidget](#) [method](#)), [271](#)
[update_plot\(\)](#) ([mesmerize.plotting.SpaceMapWidget](#) [method](#)), [275](#)
[update_plot\(\)](#) ([mesmerize.plotting.widgets.base._AbstractBasePlotWidget](#)
[method](#)), [257](#)
[update_plot\(\)](#) ([mesmerize.plotting.widgets.base.BasePlotWidget](#)
[method](#)), [258](#)
[update_roi_defs_from_configuration\(\)](#)
[\(mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList](#)
[method](#)), [214](#)
[update_workEnv\(\)](#) ([mesmerize.viewer.core.ViewerUtils](#) [method](#)), [202](#)
[use_open_dir_dialog\(\)](#) (in module [mesmerize.common.qdialogs](#)), [198](#)
[use_open_file_dialog\(\)](#) (in module [mesmerize.common.qdialogs](#)), [197](#)

`use_save_file_dialog()` (in module *mesmerize.common.qdialogs*), 198

V

`vi` (*mesmerize.viewer.modules.roi_manager_modules.roi_list.ROIList* attribute), 212

`viewer` (*mesmerize.viewer.core.ViewerUtils* attribute), 202

`ViewerUtils` (class in *mesmerize.viewer.core*), 201

`ViewerWorkEnv` (class in *mesmerize.viewer.core*), 199

`ViewHistory` (class in *mesmerize.pyqtgraphCore.flowchart.library.Data*), 248

`ViewTransmission` (class in *mesmerize.pyqtgraphCore.flowchart.library.Data*), 248

`VolCNMF` (class in *mesmerize.viewer.modules.roi_manager_modules.roi_types*), 223

W

`WidgetRegistry` (class in *mesmerize.plotting.utils*), 255

`work_env` (*mesmerize.viewer.core.ViewerUtils* attribute), 202

X

`XpowerY` (class in *mesmerize.pyqtgraphCore.flowchart.library.Math*), 251

Y

`y_pred()` (*mesmerize.plotting.KShapeWidget* property), 265

Z

`ZScore` (class in *mesmerize.pyqtgraphCore.flowchart.library.Math*), 251