Scintillate: getting started

Here we provide some basic guidance for using scintillate.

The content here is based on the release version 1.0.0 (build 20160607) accompanying the manuscript.

Scintillate: an open-source graphical viewer for time-series calcium imaging evaluation and pre-processing.

Dublon, I.A.N., Nilsson, M., Balkenius, A., Anderson, P. and Larsson M.C.

Journal of Neuroscience Methods

July 2016, the authors. Sveriges lantbruksuniversitet, Alnarp.

ian.dublon at slu.se | ian.dublon at gmail.com

What is scintillate?

Scintillate is an open-source graphical viewer for time-series calcium imaging evaluation and pre-processing. Written in Matlab, it is designed from the outset to be compiled into a platform independent self-contained application that may be run alongside bespoke image acquisition software. It provides a set of functions that help the microscope operator rapidly gauge the value of the sample being viewed, assessing the viability of the preparation in seconds, sparing microscope time, lamp hours and reducing distress to the subject. Scintillate allows pixel-by-pixel change to be detected across the whole image matrix helping to ensure that any unexpected, but real signal changes are discovered. In addition, a range of other methods of detecting and visualising change are provided.

Scintillate is not intended to replace pre- or post-processing environments but is instead designed to complement existing acquisition and pre-processing systems. For example, once signal change has been pinpointed, the user may centre the image over that specific area, change objectives, alter camera settings such as sensor binning or simply adjust regions of interest in the acquisition software.

Where can I obtain scintillate?

Scintillate is downloadable as source from GitHub at:

https://github.com/dublon/scintillate

Hardware requirements

Scintillate requires an Intel® Core™ i5 (or similar x86 architecture) or later.

Dependencies

Matlab version

Scintillate was written using Matlab 8.4 (R2014b) and therefore is known to work on this version.

Image processing toolbox

Scintillate requires the image processing toolbox.

Statistics and Machine Learning Toolbox

Scintillate uses two functions from this toolbox. They are `quantile` and `prctile`.

3rd party functions

Scintillate uses several invaluable third party functions which are included with the downloadable zip file. See licence.md for acknowledgements and licensing details.

Optional useful things

If you have the curve fitting toolbox you may substitute the use of the `fastsmooth` function with the smooth function.

The Matlab Compiler toolbox is helpful here because it easily allows you to compile your Matlab script into a standalone application. See Compiling for more information.

Supported file formats

Scintillate (release version 1.0.0) works with stacked tiff files. This decision was a pragmatic one because this is the format produced by the imaging software in our lab. Support for additional formats can easily be added programmatically and we hope that by hosting on GitHub individuals will be able to rapidly incorporate additional formats in new versions.

Compiling (optional)

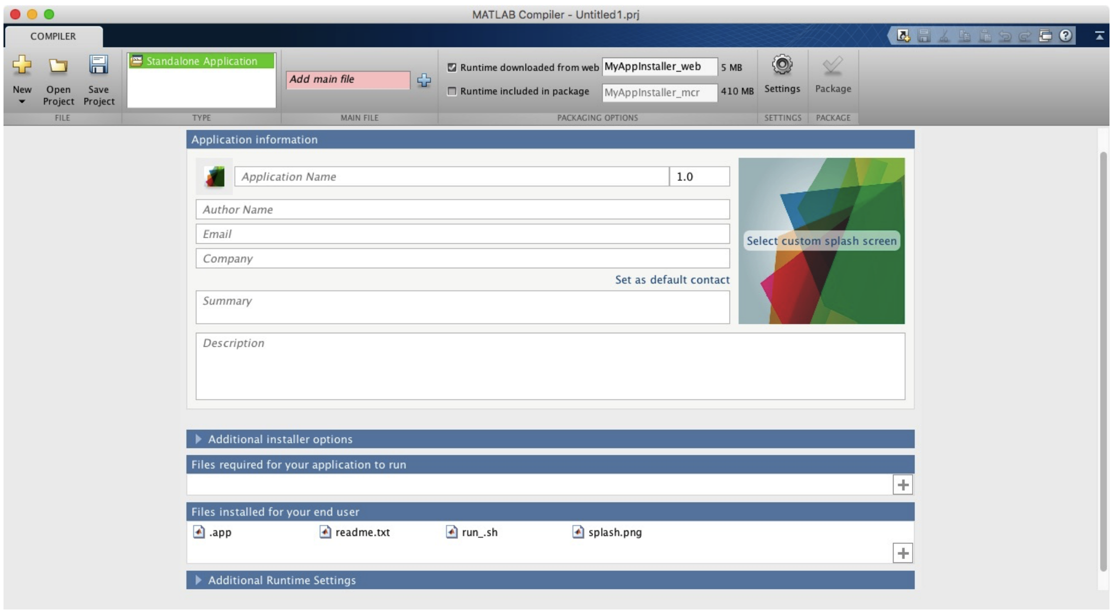
If you have the Matlab compiler toolbox it is easy to make scintillate into a standalone Mac or PC application. This means it can reside on the same workstation where you acquire data without the presence of a Matlab install or licence seat.

To do this with the release version (or with your customised version) go to the Matlab command line and type:

```
deploytool
```

and select 'Application compiler'.

You will then be presented with a dialog that looks like Figure 1.



☑ Runtime downloaded from web MyAppInstaller_web 5 MB
☐ Runtime included in package MyAppInstaller_mcr 410 MB

Application Name 1.0

Files installed for your end user: .app readme.txt run_.sh splash.png

Figure 1 Matlab compiler configuration (R2014b)

You will be asked to add the main file under 'Add main file'. Choose `scintillate.m` and click the '+' under files required for your application to run. Here add the `@TIFFStack` and `FastICA_25` folders then choose all additional `.m`, `.fig` and `.pdf` files in the directory. These should be:

```
fastica.m
fastsmooth.m
fpica.m
inputdlg_no_cancel.m
pcamat.m
plot_ica_analysis.m
remmean.m
saveastiff.m
scintillate.fig
whitenv.m
getting_started.pdf
```

Under packaging options, you can decide if you will include the Matlab Compiler Runtime libraries or if your installer will instead download them at install time from the Mathworks website. You may also adjust configurations here depending on requirements though in principle things should now be ready to package. Select package and this will build an installer package for your self-contained executable.
Please note, on Windows systems (tested with Windows 7 and above) installation of the application requires an Administrator account.

Getting started from within Matlab

Unpack your zip file and from within Matlab navigate to the working directory containing the unpacked file.

Execute:

```
scintillate
```

Scintillate will then perform a small amount of path housekeeping and will add to Matlab the path to `@TIFFStack`. A graphical window will be created. You will be presented with a GUI that should look like Figure 1:

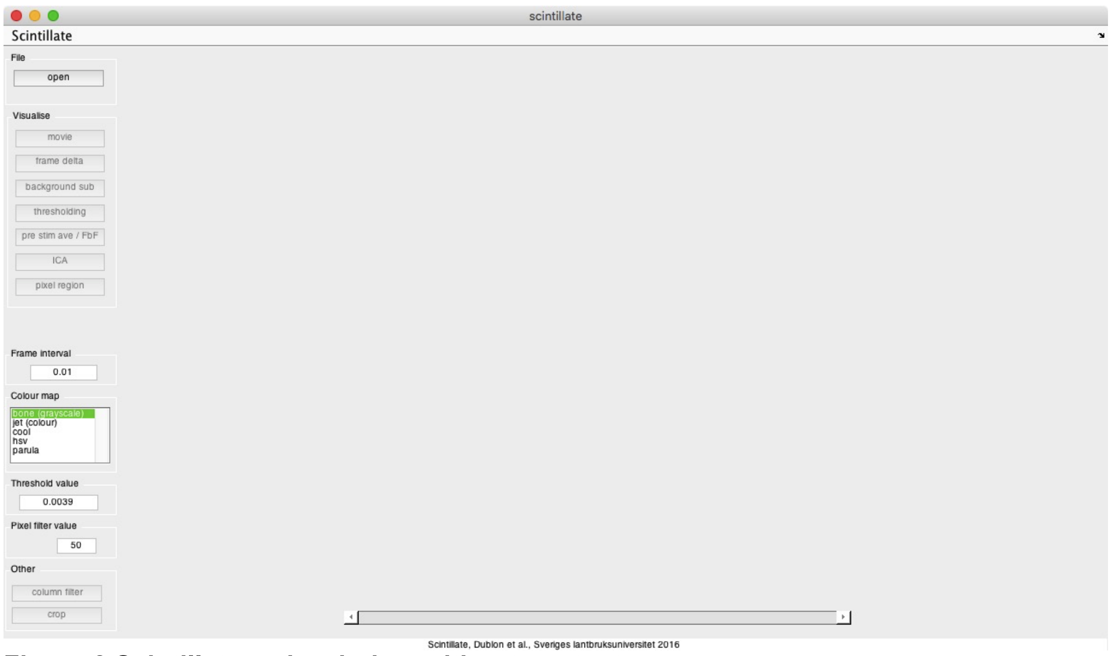


Figure 2 Scintillate main window without content

As is visible a main window has been created with buttons on the left, a slider at the base, a text status area below the slider and a menubar with a single 'Scintillate' menu item.

Customising

All code is commented and all changes can be made in one of two files: `scintillate.m` and `plot_ica_analysis.m`. It should be simple to edit and adjust according to your imaging requirements. It is also possible to edit scintillate.fig from within Matlab's graphical integrated development environment (`guide`). This allows you to easily customise the GUI interface parameters.

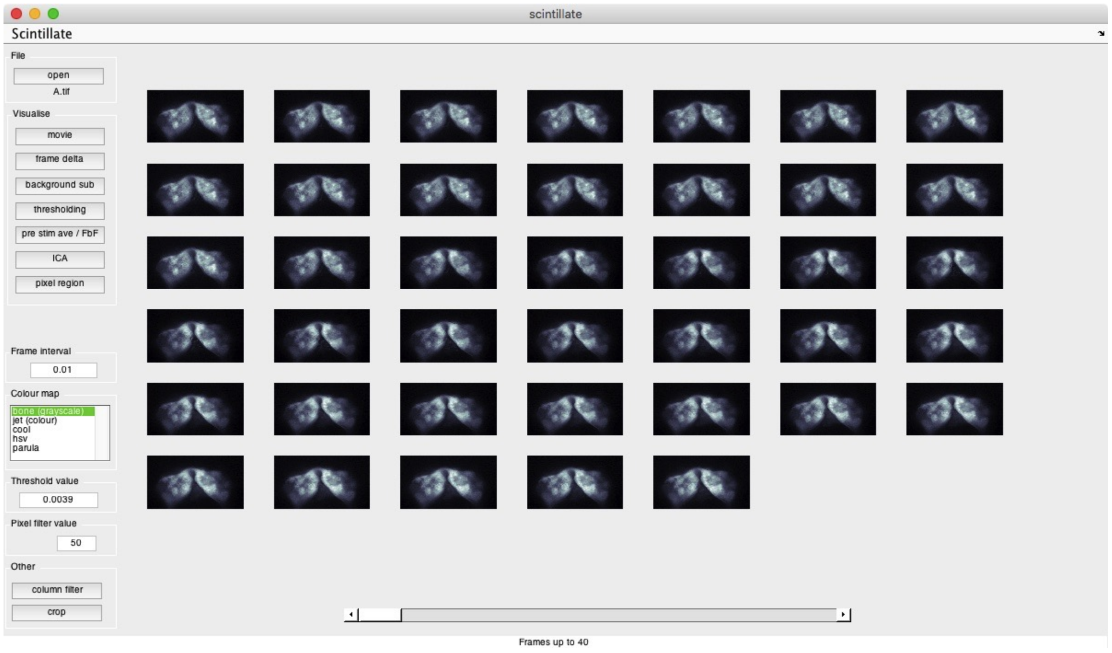
Opening data

Clicking open will bring up a `uigetfile` based dialog box that should mirror the look and feel of your operating system. At present scintillate only supports stacked tiff data.

Once file loading has been initiated you will be prompted to enter the frame that the stimulus is delivered to the preparation. Defaults are frame 11 though this is easily changed in the source.

Computations and initial renderings

Scintillate will use the 3rd party `TIFFStack` to efficiently store the data in a memory efficient way. Please see the experimental part of the accompanying paper for more information. It will provide an overview by outputting a thumbnail of each of the frames provided (Figure 3). Additionally, a second child window will be spawned with an average of all the frames prior to the stimulation (Figure 4). This provides a really quick way to view the file.



Frame interval 0.01

Threshold value 0.0039

Pixel filter value 50

Figure 3 Scintillate with open data, in this case a 40 framed file named A.tif

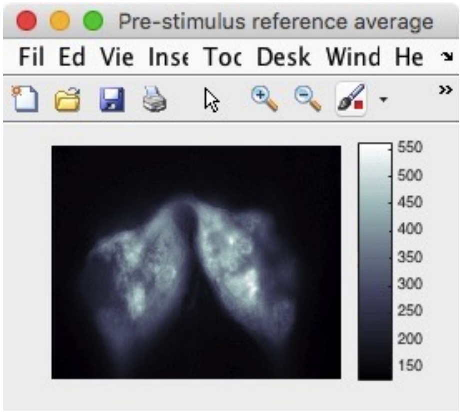

Pre-stimulus reference average
Fil Ed Vie Inse Toc Desk Wind He
550
500
450
400
350
300
250
200
150

Figure 4 Scintillate with open data, in this case the pre-stimulus average for A.tif

The open filename will be presented underneath the open button. Some buttons previously inactivated will become active. The status bar text will be updated with progress information. The variable `tsStack` will be created containing the read in data. `tsStack` is passed to other functions in the script using `guidata`.

The panel

A panel on the left provides access to buttons with respective functions. It is split into 7 parts. Each part has a tooltip providing user information that is accessible when you hover the mouse over it.

File

Here with the open button, and the status text.

Visualise

For viewing differences:

- via playback of each frame (movie);
- looking at the difference between each frame (frame delta);
- performing pre-stimulus background subtraction (background sub);
- performing thresholding (thresholding);
- producing a new window with pre-stimulus average and generating frame-by-frame output (pre stim ave / FbF);
- performing Independent Component Analysis (ICA);
- via the ability to magnify individual pixel regions (pixel region).

Frame interval

Here you can set the pause time between frame iterations in the frame playback loop.

Colour map

Here you can select which colour map you wish to use to view the image.

Threshold value

Here you can specify what value you would like to use with the thresholding. Level is a normalized intensity value that lies in the range 0 - 1. If 0 is provided an automatic value will be generated. This can be a good starting point.

Pixel filter value

Here you can specify the filter size in pixels to use for the pixel filtering used in the thresholding. Scintillate uses `bwareaopen` to remove objects that have fewer than the specified number of pixels.

Column filter button

Here you can process your image through a column filter (`colfilt`) and view the output as a movie. Once clicked the user is asked for the size in pixels of the filter.

Crop

Here you can crop and export parts of your stacked tiff as a new stacked tiff. These can then be directly read in again as required or viewed in any other image processing software. This may be especially useful in instances where you can clearly see autonomic movement at the periphery of the image and may wish to simply crop these out.

The menu bar

A menu bar provides the options to read licensing information, gain information about the program and launch this 'getting started' .pdf. Selecting quit will quit the standalone application (if compiled) or exit Matlab.

The status bar

At the base of the screen there is a status bar that will display pertinent information.

The slider

The slider bar at the base of the main window may be used to go back and forth between frames and should be updated with frame playback progress.

Visualising options

Here we provide some more information on the possible ways to view change.

Movie

Here things are at their most simple where each frame is presented sequentially with the delay between frames set using Frame Interval. The slider will mirror the progress of the frame playback.

Frame delta

Here scintillate uses `imabsdiff` to look at the absolute difference between each frame and display that. This will provide a frame-by-frame change counting up when first clicked, and counting down when clicked again. The slider will mirror the progress of playback.

Background sub(traction)

Here scintillate will prompt for the stimulus delivery frame and provide an average of frames prior to this point. This pre-stimulus average will then be subtracted from every subsequent frame afterwards.

Thresholding

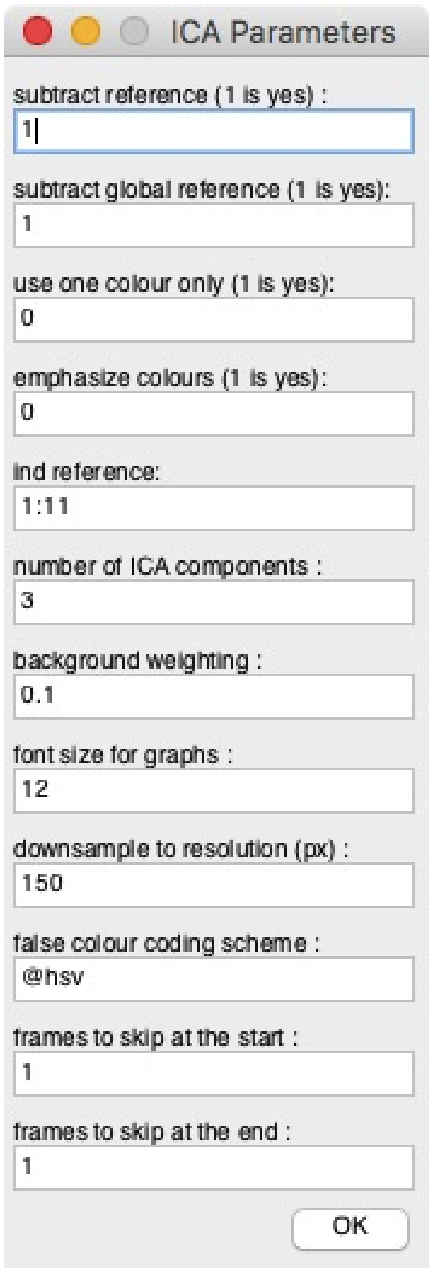
Here depending on the values given for pixel filter and threshold value scintillate will convert an intensity image to a binary image using `im2bw`.

Pre stim ave(rage) /FbF (frame by frame)

Because scintillate displays most visualisations on the main window, destroying any previous content it may be useful to recreate frame-by-frame output. Pressing this button will provide a new window with the same recreated frame-by-frame output. Again the pre-stimulus average is also presented.

ICA

Here it is possible to perform independent component analysis courtesy of the `FastICA` toolbox (see license.md for more information). When clicking this button, the user is presented with a dialogue where ICA variables may be adjusted. See figure 5 and box 1.



subtract reference (1 is yes) : 1

subtract global reference (1 is yes): 1

use one colour only (1 is yes): 0

emphasize colours (1 is yes): 0

ind reference: 1:11

number of ICA components : 3

background weighting : 0.1

font size for graphs : 12

downsample to resolution (px) : 150

false colour coding scheme : @hsv

frames to skip at the start : 1

frames to skip at the end : 1

Figure 5 Independent Component Analysis variable dialogue

Box 1: ICA parameters

Defaults can be edited in the source code.
Subtract reference:
Yes = 1 No = 0. Default: 1. Removes background signal.

Subtract global reference:
Yes = 1 No = 0. Here you can choose to subtract the global signal mean. Default: 1.

Use one colour only:
Yes = 1 No = 0. If yes, each pixel will show the response from at most one component.

Emphasize colours:
Yes = 1 No = 0. Here you can make the colouring stronger.

Ind reference:
A range expressed as start:end (e.g. 1:11). Images in this range will be used to generate the background image.

Number of ICA components:
Here you can select the number of components you wish to attempt to separate.

Background weighting:
A value between 0 and 1 that will determine the weighting of the background versus the component-colour image.

Font size for graphs:
Here you can alter the font size for the graphs.

Downsample to resolution:
Here you can choose to downsample the image to a specific size (provided in pixels).

False colour coding scheme:
Here you can choose which colour scheme to use. Choices relate to Matlab standard colour palettes.

Frames to skip at the start:
Frames to exclude at the start of the tiffstack.

Frames to skip at the end:
Frames to exclude at the end of the tiffstack.

Once these variables have been accepted scintillate will perform the ICA and produce an output window. See Figure 6 and box 2.

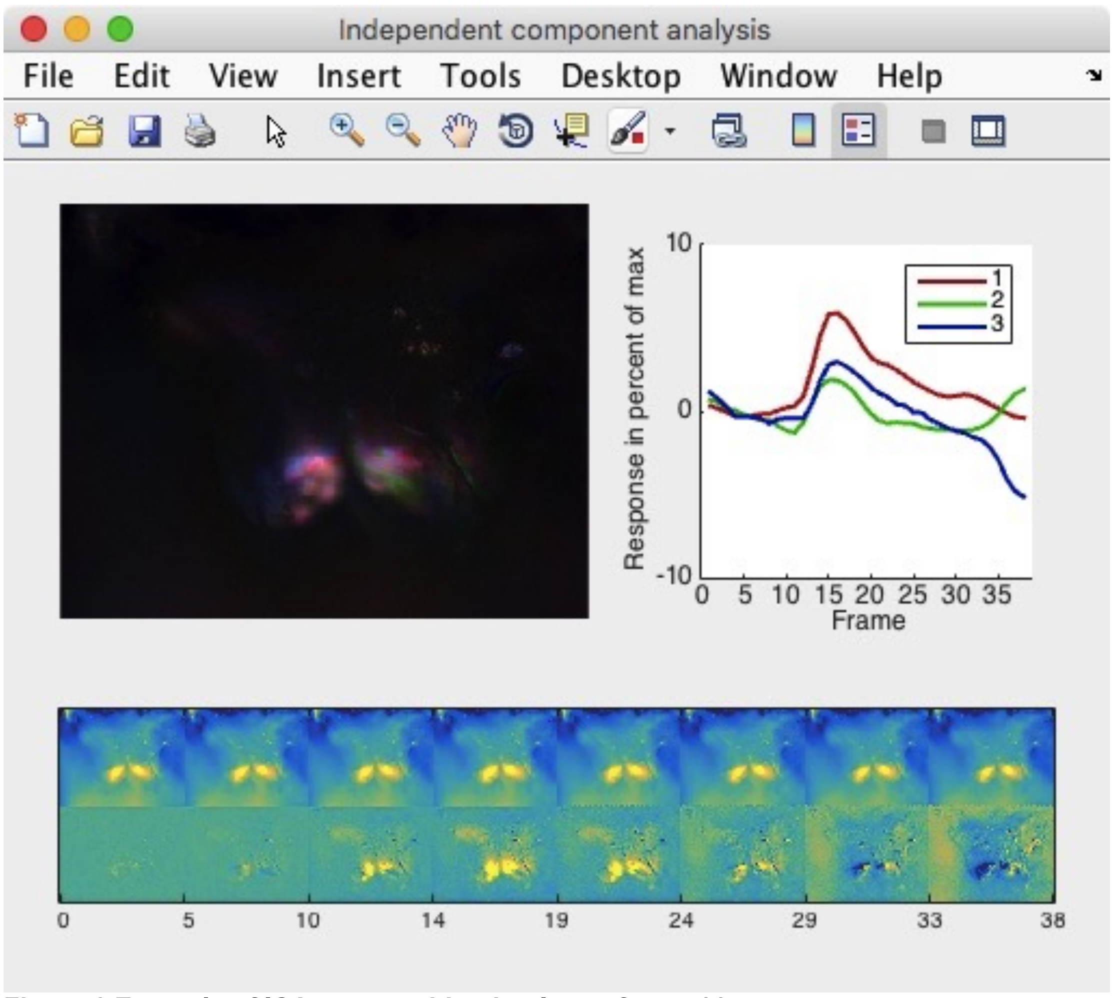


Figure 6 Example of ICA output with stimulus at frame 11.

Box 2: ICA output window
Here ICA output is split into three sections

Top left: shows false colour coded image relating to ICA components
Top right: shows the three most pronounced independent components (here three signals are requested, and three are separated and shown).
Base: shows snapshots of the input images for selected frames at the top, and the pre-processed images used for the ICA analysis at the bottom

The menu bar here is the standard menu bar though the print button has been substituted with Matlab's `printpreview` function to allow for .pdf output and customisation.

Pixel region

Here you can inspect specific regions of your image by magnifying.

What methods to use and in which circumstances

Scintillate provides a set of tools to rapidly establish the worth of the preparation. This should help with deciding whether to continue to image or stop and start afresh. When answering the question, “is my preparation viable” it is possible that some tools are more useful than others though it very much depends on the preparation itself.

When some change is visible but you need to further pinpoint or confirm the location

In this case frame delta and background subtraction are a good place to start. In such instances whole matrix pre-stimulus subtraction from post stimulus frames (pre-stimulus background subtraction) may provide a clearly delineated area of response.

When there is really no instantly visible change

The best starting option is likely ICA. The ICA method can decompose a complex signal into multiple simple constituent signals. We thus employ ICA as a convenient method for detecting and visualising source signals, in order to rapidly provide an overview of typical signal patterns and map their spatial positions. We typically expect one or two signals from the region we are interested in. Other signals that may appear can be related to motion, and will typically result in high intensity along the edges of the object. ICA can in the first instance be run with default settings.

Questions / comments

We hope that you find scintillate useful in your imaging work and naturally we welcome feedback and active development at GitHub.

Ian Dublon and Markus Nilsson July 2016.