MESOSCALE BRAIN EXPLORER, A FLEXIBLE PYTHON-BASED IMAGE ANALYSIS AND VISUALIZATION TOOL

by

Cornelis Dirk Haupt

B.Sc, University of British Columbia, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Neuroscience)

The University of British Columbia
(Vancouver)

August 2017

Abstract

Imaging of mesoscale brain activity is used to map interactions between brain regions. This work has benefited from the pioneering studies of Grinvald et al., who employed optical methods to image brain function by exploiting the properties of intrinsic optical signals and small molecule voltage-sensitive dyes. Mesoscale interareal brain imaging techniques have been advanced by cell targeted and selective recombinant indicators of neuronal activity. Spontaneous resting state activity is often collected during mesoscale imaging to provide the basis for mapping of connectivity relationships using correlation. However, the information content of mesoscale datasets is vast and is only superficially presented in manuscripts given the need to constrain measurements to a fixed set of frequencies, regions of interest, and other parameters. We describe a new open source tool written in python, termed mesoscale brain explorer (Mesoscale Brain Explorer (MBE)), which provides an interface to process and explore these large datasets. The platform supports automated image processing pipelines with the ability to assess multiple trials and combine data from different animals. The tool provides functions for temporal filtering, averaging, and visualization of functional connectivity relations using time-dependent correlation. Here, we describe the tool and show applications, where previously published datasets were reanalyzed using MBE.

Lay Summary

We've designed and implemented a cross-platform standalone application with a user-friendly interface that performs most of the brain connectivity mapping processing steps we perform in the lab. The application also outputs interactive visualizations to explore the data. As a standardized approach that automates processing steps, saves user input where possible and is easy to install, the application makes it easier for neurophotonic researchers without programming skills to perform the analysis they want to immediately. The application is designed according to common software engineering principles so that it can be easily modified and its functionality expanded by someone with an undergraduate level of programming experience. A step-by-step tutorial is provided in this regard.

Preface

I wrote chapters 1,3 and 4 and am responsible for all figures unless the figure caption states the figure was adapted from a source. Chapter 2 and Appendix A.1 consist of an unedited manuscript previously published - **Haupt, Dirk, Matthieu P. Vanni, Federico Bolanos, Catalin Mitelut, Jeffrey M. LeDue, and Tim H. Murphy. "Mesoscale Brain Explorer, a Flexible Python-Based Image Analysis and Visualization Tool." Neurophotonics 4, no. 3 (July 2017): 031210. doi:10.1117/1.NPh.4.3.031210.** . Dr. Timothy H. Murphy supervised the project, worked with me to write the published manuscript, and provided financial support.

An initial prototype and framework was implemented in collaboration with a paid consultant, Kristian Csepej. Much of MBE's framework implemented by Kristian is itself built off of an open-source project I found on Github which was implemented by Micheal Hogg for bone mass density analysis. Code was taken from Jeffrey LeDue, Fedrico Bolanos and Catalin Mitelut and included in the backend. I am responsible for the majority of the design, implementation and testing that went into MBE. Matthieu Vanni taught me most of the underlying neurophotonic principles and provided design guidance. Blair Jovellar acted as the application's primary user and guided the application's development through constant feedback and bug-finding.

Table of Contents

List of Tables

List of Figures

Glossary

AAV Adeno-Associated Virus

AC Anterior Cingulate

AC Anterior Cingulate

AP Action Potential

BAMS Brain Architecture Knowledge Management System

BC Barrel Cortex

BN Bayesian Network

CCA Cross-correlation Analysis

CHR2 Channelrhodopsin-2

CSV Comma Seperated Value

DBN Dynamic Bayesian Network

DCM Dynamic Causal Modelling

DOI Document Object Identifier (seehttp://doi.org)

DRY Don't Repeat Yourself

EVO Evolutionary Development Model

FLA Front-Line Analyst - primary user of MBE

FMRI Functional Magnetic Resonance Imaging

FRET Forster Resonance Energy Transfer

GC Granger Causality

GECI Genetically Encoded Calcium Indicator

GEI Genetically Encoded Indicator

GEPI Genetically Encoded pH Indicator

GETI Genetically Encoded Transmitter Indicator

GEVI Genetically Encoded Voltage Indicator

GSR Global Signal Regression

GUI Graphical User Interface

HL Hindlimb Cortex

ICA Independent Component Analysis

IDE Integrated Development Environment

IOS Intrinsic Optical Signal

JBSS Joint Blind Source Separation

JSON JavaScript Object Notation

LASSO Last Absolute Shrinkage and Selection Operator

LSI Laser Speckle Imaging

M1 Primary Motor Cortex

M2 Secondary Motor Cortex

MBE Mesoscale Brain Explorer

MI Mutual Information

MM MilliMeter

PCA Principal Component Analysis

ROI Region of Interest

RS Retrosplenial Cortex

SEM Structural Equation Modelling

SEP Super Ecliptic pHluorin

SPC Seed Pixel Correlation

STDEV STandard DEViation

UI User Interface

V1 Visual Cortex

VSD Voltage Sensitive Dye

Acknowledgments

The development of MBE was fraught with long periods of no tangible results. I thus first and foremost thank my supervisor Timothy Murphy for providing me with support and the space I needed to push through to the end and learn how to develop the application and finally bring it into a usable state. I have grown tremendously as a developer and a researcher as a result of your patience, support and feedback

Thank you to my committee, Tamara Munzner, John Steves, and Nick Swindale for your guidance and continued interest in my work. I especially thank Tamara for providing guidance on the literature of reproducible software design in academia, Nick for providing guidance on the the literature of the various brain connectivity methods relevant to MBE and finally John for getting me thoroughly into Neuroscience research in the first place. I owe you all an enormous debt.

I also owe a special and tremendous thank you to my colleague Blair Jovellar who was the primary user and tester of the application as it was proceeding through its earlier and more bug-ridden stages. The frustration she endured led to invaluable feedback that now ensures that any and all future users of the application will not have to go through the same woes.

Thank you to the all the members of the Murphy Lab that have helped me throughout this project, and otherwise made me feel like I was part of the Murphy Lab family.

Finally, I would like to offer my deepest thanks to my friends and parents for their love, encouragement and never-ending support.

Dedication

到石仪和郁海菲。永恒的记忆和令人兴奋的未来。爱与爱

Chapter 1

Introduction

1.1 Functional Mesoscale Mapping

Connectomics refers to a branch of biotechnology concerned with applying the techniques of computer assisted image acquisition and analysis to the structural mapping of sets of neural circuits or the complete nervous system of selected organisms using high-speed methods, organizing the results in databases (Lichtman and Sanes 2008). Perturbations in the neural networks that comprise a connectome are associated with a wide range of clinical problems affecting the nervous system (Lichtman and Sanes 2008). Understanding the relationship between anatomical links (structural map) and statistical dependencies (functional map) within the brain is essential to understanding the large-scale reorganization of neuronal networks that occurs at multiple levels following pathologies such as stroke or depression (Xerri 2012; Osten and Margrie 2013; Nudo 2013). Structural mapping studies have been vital to shaping our understanding of how the brain is wired, thereby providing a framework that constrains functional connectivity (Oh et al. 2014). With analogy to geographic maps, structural mapping informs us of where the roads are, but functional mapping is required to tell us how much traffic is expected on which roads given which conditions (Dance 2015). Therefore functional connectivity contrasts with structural connectivity in that there is no single map. Behavioural states observed in animals under study, the level of anaesthesia and other environmental stimuli will affect apparent functional connections (Petersen et al. 2003; Crochet and Petersen 2006). Given this variation and potential for confounds, optogenetics - the use of light and genetic manipulation to control and monitor biological systems at the neuronal-population level has proven revolutionary (Boyden et al. 2005; T.-W. Chen et al. 2013; A. Miyawaki et al. 1997; Deisseroth 2011).

Connectomics can roughly be defined at three different levels of scale, corresponding to levels of spatial resolution in brain imaging: micro-, macro- or mesoscale (Kötter 2007; Sporns 2010; Bohland et al. 2009). At the microscale, connectivity is described at the level of individual synapses and mapping is done neuron-by-neuron. At the time of writing, mapping at this scales amounts to enormous time and resources due to the number of neurons comprising a brain easily ranging into the billions in highly evolved organisms (Sporns 2010). Microscale approaches are thus best suited for relatively small volumes of tissue ($<$1mm (Oh et al. 2014; Sporns 2010; Bohland et al. 2009)). At the macroscale, large brain systems are parcellated into

anatomically distinct modules based on distinct patterns of activity, and long-range, region-to-region connections are inferred from white matter fibre imaging techniques such as diffusion tensor imaging a living brain. This is far from cellular-level resolution, with sizes of single volume elements (voxels) typically exceeding a MilliMeter (MM) and therefore exceeding the entire volume of a typical microscale map (voxels >1MM (Oh et al. 2014; Sporns 2010; Bohland et al. 2009)). Clearly an intermediate "mesoscale" with spatial resolution of hundreds of micrometers is needed to capture anatomically and/or functionally distinct neuronal populations formed by local circuits that link hundreds or thousands of individual neurons, thereby bridging the vast divide between micro and macroscale (Sporns 2010; Swanson and Lichtman 2016). This intermediate scale presents major technical challenges and at this time can only be probed on a small scale with invasive neuroanatomical tracer techniques that enable whole-brain mapping or very high field magnetic resonance imaging on a local scale (Oh et al. 2014). None of the leading neuroanatomists of the nineteenth and early twentieth century attempted a synthetic, global wiring diagram model of the vertebrate nervous system at the mesoconnection level (Swanson and Lichtman 2016). The only example of such an attempt from these eras is Herrick (1948)'s description of the tiger salamander's brain based on 50 years of personal research. As such, the mesoscale represents the scale where our knowledge is arguably most incomplete, rivalled only with the nanoscale that considers connections not between neurons as in the microscale, but at individual synapses (Swanson and Lichtman 2016). Rapid progress is currently being made on fly mesoscale mapping using a combination of genetic, imaging, and neuroinformatics technologies (Chiang et al. 2011; Shih et al. 2015; Wolff, Iyer, and Rubin 2015). Researchers have also made promising starts on mouse whole-brain macroscale mapping (Zingg et al. 2014; Oh et al. 2014), which can readily be extended to the mesoconnectome level as a high level framework (Swanson and Lichtman 2016). Moreover, as a scale fundamentally concerned with the connectivity of local neuronal populations, cell-type-specific mesoscale projects also exist as cell types are fundamental cellular units often conserved across species. Consequently, one of the biggest hurdles to mesoscale mapping is combining data from both whole brain imaging and cellular imaging to attain a unified mesoscale map of how specific neuronal populations are connected across the brain. Central to such unification is polythetic classification (Swanson and Lichtman 2016), in which all relevant neuron data are evaluated statistically with multiparametric methods such as principal component and cluster analysis so as to form distinguishable clusters in parametric space for neuron classification (Bota and Swanson 2007).

The unique challenge posed by mesoscale mapping has led to the development of big science high-throughput structural brain mapping projects. As of 2014 three projects have emerged that inject anterograde and/or retrograde Adeno-Associated Virus (AAV) vectors expressing fluorescent proteins according to the mapped animals' stereotaxic coordinates. Fluorescent protein labelling of all projects are subsequently imaged using a systematic high-throughput serial tomography system to construct three-dimensional maps of the neuronal axons. Two of these projects, the Allen Mouse Brain Connectivity Atlas (Oh et al. 2014) and the Mouse Connectome Project (Zingg et al. 2014) both aim to map the mesoscale connections in the mouse brain. The Brain Mapping by Integrated Neurotechnologies for Disease Studies (Brain/MINDS) (Okano, Atsushi Miyawaki, and Kasai 2015) in contrast aims to map the marmoset brain at micro, macro and mesoscale. A fourth and final mesoscale mapping project that emerged in 2014 is the Rat Neurome Project (Swanson,

Sporns, and Hahn 2016; Bota, Sporns, and Swanson 2015) that aims to build a full mesoscale connectome of the rat brain through systematic curation of the primary neuroanatomical literature, annotating and collating over 70,000 rat cortical association mesoconnection reports for storage in the Brain Architecture Knowledge Management System (BAMS) (Bota, Sporns, and Swanson 2015). Contemporary structural neuroscience suffers from relatively chaotic nomenclature that obscures communication and is disastrous for formal network analysis and knowledge management systems. BAMS therefore makes use of a defined set of terms and relationships between them (Swanson and Bota 2010), along with a human- and machine-readable formal modelling language for neural connectivity (Swanson and Lichtman 2016; R. A. Brown and Swanson 2013; Swanson and Lichtman 2016). However, these projects provide static, structural connectivity maps. Although necessary, these maps are insufficient for understanding function. Understanding functional connectivity in a living brain requires fundamentally different approaches (Oh et al. 2014).

Functional brain mapping technologies have been greatly refined since the days of Fritsch and Hitzig (1870) who used wires and batteries to stimulate the cortex. Penfield and Boldrey (1937) developed hand-held electrical probes to stimulate the cortical surface during surgery in epileptic patients that he used to define the cortical location of the motor cortex. Asanuma, Arnold, and Zarzecki (1976) developed intracortical microstimulation, which involves lowering electrodes into the cortex and passing current in order to drive cells in a region of interest. Subsequently, surface stimulation with electrode arrays was developed for use in rodents (Hosp et al. 2008). However, electrode-based brain stimulation methods have common disadvantages. Neuronal sub-types cannot be selectively activated and instead axons are indiscriminately activated. Some neuron types also resist extracellular electrical recordings owing to unfavourable cell morphology, weak electrical dipoles, or the organization of the extracellular tissue (Michael Z. Lin and Schnitzer 2016). Because of sampling bias for active cells, extracellular recordings typically overestimate rates of Action Potential (AP)s, or spikes (Michael Z. Lin and Schnitzer 2016). Moreover, some degree of damage is caused where the electrode is implanted causing concern that mapping results may be confounded by the damage caused.

The advent of optogenetics (Boyden et al. 2005; T.-W. Chen et al. 2013; A. Miyawaki et al. 1997; Deisseroth 2011) has made it possible to stimulate neuronal types selectively using light energy, either by uncaging neurotransmitters (Shepherd, Pologruto, and Svoboda 2003; Luo, Callaway, and Svoboda 2008) or directly activating genetically targeted light-sensitive channels (Daniel Huber et al. 2008; F. Zhang, Aravanis, et al. 2007). A key advantage of optogenetics is that light-sensitive proteins (opsins) can be activated locally for investigation of connectivity with spatial resolution on the order of single neurons. Optogenetic methods for functional mapping have thus been applied in experiments ranging from *in vitro* investigation of microscale circuitry, to *in vivo* investigation of mesoscale cortical connectivity (Lim, J. LeDue, et al. 2013). Chief among the light-sensitive channels used for optogenetics is Channelrhodopsin-2 (CHR2) (Georg Nagel et al. 2003), a non-selective cation channel that opens in response to blue light that can be expressed selectively using viral vectors, in utero electroporation or as is commonly used, through the creation of transgenic mice lines (F. Zhang, L.-P. Wang, et al. 2006; Arenkiel et al. 2007; Petreanu et al. 2007; Daniel Huber et al. 2008; Kuhlman and Huang 2008; Ting and Feng 2013). CHR2-expressing brain regions can be systematically photo-stimulated and functional connectivity mapped by recording subsequent changes in organic

fluorescent compounds or whole-cell currents (Lim, J. LeDue, et al. 2013). CHR2-expressing subpopulations of neurons can be used to map functional activity in vitro (Boyden et al. 2005; X. Li et al. 2005; Ishizuka et al. 2006), in slice preparations (Ishizuka et al. 2006) and *in vivo* (X. Li et al. 2005; G. Nagel et al. 2005). For its precision in being able to target and activate specific neuron types in discrete brain regions with millisecond temporal resolution, with reliable action potential generation up to ~30Hz that induce synaptic transmission, CHR2 is essentially the neurophotonic researcher's "scalpel" (Boyden et al. 2005; X. Li et al. 2005). As an invasive technique however there remains concern that functional dynamics are artificially created and thus do not accurately mimic typical biological conditions. For example, CHR2 stimulation may evoke relatively larger calcium transients compared to current injection, increasing the probability of neurotransmitter release (Schoenenberger, Schärer, and Oertner 2011). Over-expression of CHR2 on particular neurons may also significantly affect normal neuronal function (Miyashita et al. 2013). Finally, since animals typically need to be restrained during this lengthy procedure, there is concern that measured brain activity do not accurately mimic what brain activity would look like were the mouse in its natural environment.

1.2 Brain Spontaneous Activity Measuring Techniques

The solution to the ever-present problem of invasiveness has been to instead collect large spontaneous brain activity datasets without explicitly eliciting any unnatural stimuli, while simultaneously monitoring the animal's behaviour longitudinally so as to link patterns of spontaneous activity with particular patterns of behaviour (Wiltschko et al. 2015). An exciting application of this approach will be to trace the synaptic circuits of neurons functionally characterized in head-fixed behaving animals engaged in tasks related to spatial navigation, sensorimotor integration and other complex brain functions (Christopher D Harvey et al. 2009; Christopher D. Harvey, Coen, and Tank 2012; D. Huber et al. 2012). With large datasets of spontaneous brain activity, how a damaged functional connectome changes following a therapeutic intervention can be more accurately modelled.

The accuracy of any particular measurement involving fundamental, quantum interactions such as counting photons is limited by Poisson statistics (Pawley 2006). An analysis of Poisson statistics is well beyond the scope of this thesis, but the crux is that this uncertainty associated with counting photons captured during optical imaging is introduced by no fault of the equipment or experimenter and is merely a property of quantum behaviour. Such uncertainty is therefore referred to as the source of intrinsic noise. Extrinsic noise, by contrast, is introduced accidentally by poor techniques or lacking technology. Just as the source of noise in optical imaging can be categorized into intrinsic and extrinsic, so too can optical imaging techniques be classified into either category. Extrinsic optical brain imaging techniques (See Fig. 1.1) involve introducing an external substance into the brain to act as a proxy for the biological phenomenon being measured by enhancing its signal. Extrinsic optical brain imaging techniques covered can further be divided into organic or genetic techniques. Organic extrinsic techniques typically involve the introduction of an organic fluorescent dye sensitive to the activity of a biological phenomenon under investigation. Voltage sensitive dye for instance provide linear measurements of the firing activity of single neurons. Genetic extrinsic techniques involve genetically modifying an organism to express light-sensitive ion channels that fluoresce due to brain

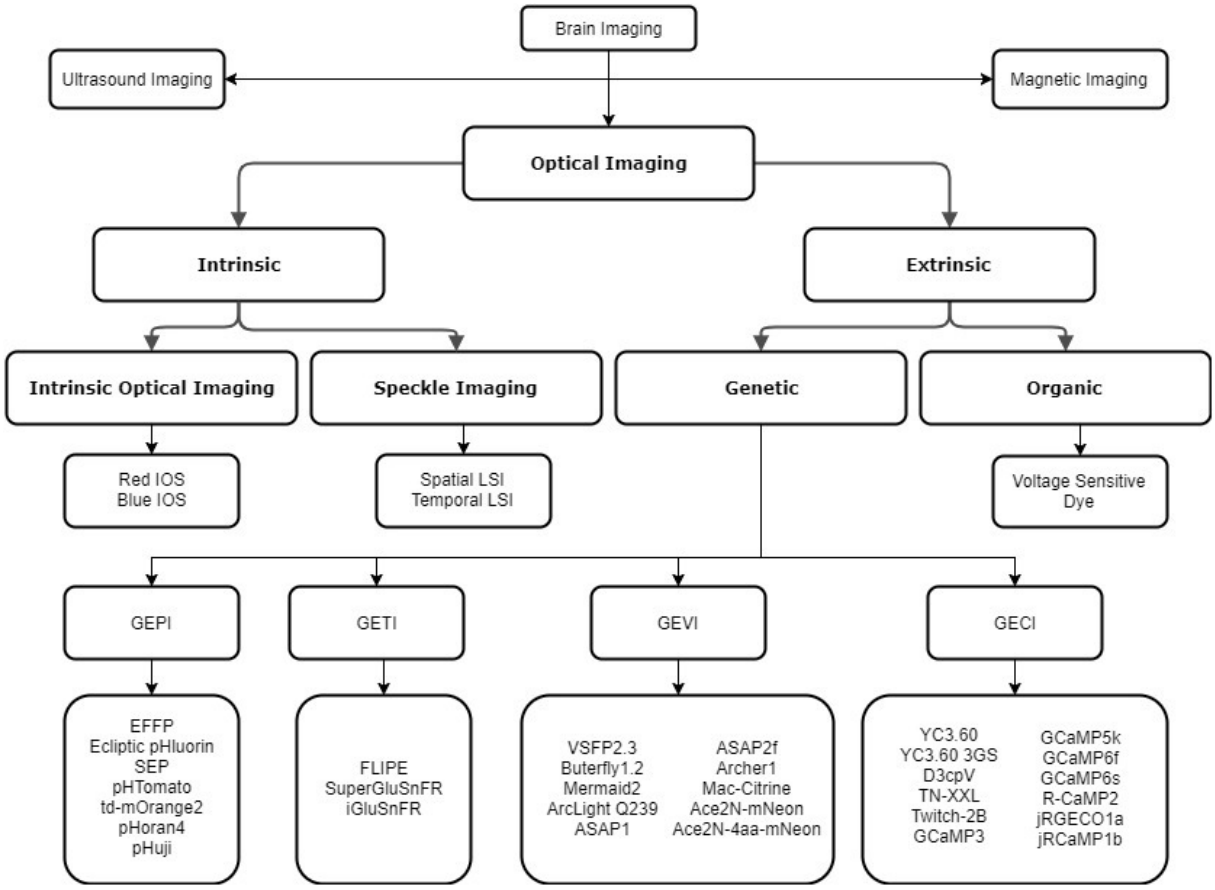


Figure 1.1: Taxonomy of optical imaging brain imaging techniques in widespread use to collect spontaneous brain activity in awake behaving animals.

activity under certain light conditions. This allows for pin-point specificity as the genome of an organism can be modified such that only a particular neural type expresses light-sensitive channels or only a particular. Upon imaging under light only this neuron type is captured. In contrast, intrinsic optical brain imaging techniques (see Fig. 1.1) simply involve using the inherent reflectance properties of the brain to record a biological phenomenon and is typically used to measure blood flow (laser speckle) and blood volume (intrinsic optical imaging).

The type of fluorophore being imaged will dictate which physical structures are labelled and what form of activity is best monitored. Genetically Encoded Calcium Indicator (GECI)s such as GCaMP6 monitor suprathreshold activity (T.-W. Chen et al. 2013). GCaMP6 can be expressed in specific cell types selectively, from all neurons in the cortex, to specific sub-types or layers, dendrites or cell bodies, sparsely or densely or even other reactive cells such as astrocytes (Heim et al. 2007; T.-W. Chen et al. 2013; Khakh and Sofroniew 2015; Akerboom, Carreras Calderón, et al. 2013). GCaMP6 can be used to spatially resolve individual neurons or provide a spatiotemporally integrated ensemble representation of neural activity in each labelled region, making it ideal for mesoscale functional mapping that requires both whole brain imaging and

cellular imaging to attain a unified mesoscale functional map (Y. Ma et al. 2016). Transgenically expressed glutamate-sensing fluorescent reporter iGluSnFR, a Genetically Encoded Transmitter Indicator (GETI), is used to characterize in vivo activity at frequency bands higher than 1-3Hz (Xie et al. 2016). Voltage Sensitive Dye (VSD) is more resistant to vascular artifacts and monitors mostly subthreshold activity (Carandini et al. 2015). As such, VSD resolves particular spatiotemporal patterns of subthreshold activity (Amiram Grinvald and Hildesheim 2004; Chan et al. 2015). VSD yields more diffuse labelling which localize to cell membranes (Shoham et al. 1999; Orbach, L. B. Cohen, and A. Grinvald 1985). Particular cellular structures can also be labelled such as through the use of flavoprotein fluorescence imaging, an intrinsic imaging technique where the fluorophore flavoprotein is used as an indicator of oxidative metabolism by localizing to mitochondria (Shibuki, Hishida, et al. 2003; Kozberg et al. 2016; Shibuki, Ono, et al. 2006). Together these mesoscopic brain imaging techniques provide powerful tools for longitudinally studying the dynamics of neural populations at mesoscopic scales noninvasively - thereby pathological models of central nervous system disease such as stroke and depression can be characterized without introducing artificial stimuli that may exacerbate confounds. These wide-field neuroimaging methods are also readily compatible with optogenetic approaches to modulate brain activity in awake animals (N. A. Scott and Timothy H. Murphy 2012). As a result, mesoscopic optical recording of neural activity across the exposed cortex has become ever more prevalent, further facilitated by greatly improved high-speed, sensitive camera technology (Majid H. Mohajerani, McVea, et al. 2010; Vanni and Timothy H. Murphy 2014; Ackman, Burbridge, and Crair 2012; Bouchard et al. 2009; Daniel et al. 2015).

Despite camera technology improvements, none of these fluorophores are without intrinsic properties that impose limitations on their signal interpretation or other concerns relating to potential sources of extrinsic noise from the experimental setup required.

1.2.1 Voltage Sensitive Dye

Wide-field optical imaging of neural activity was first demonstrated 30 years ago using cortical application of fluorescent VSDs (Orbach, L. B. Cohen, and A. Grinvald 1985). Although VSDs held great promise as direct optical indicators of neural activity, VSDs can bind to glial membranes and show slow changes in the membrane potential of activated glia (Petersen et al. 2003; Konnerth and Orkand 1986), their very fast response times and their very small $\Delta F/F_0$ ratios (typically 0.15%) make them challenging to use (Y. Ma et al. 2016), their dendritic origin of signal can affect spatial resolution of VSD to the point that a dye signal in a particular cortical site does not imply that cortical neurons at that site are generating action potentials (Malach et al. 1993), and finally the need to apply VSD to the exposed brain prior to imaging results in phototoxicity (Michael Z. Lin and Schnitzer 2016; Peterka, Takahashi, and Yuste 2011). Most of these concerns can be mitigated. The contribution of slow glial depolarization to the VSD signal is minimal and can be parsed out as glial activity reported is dissimilar to neuronal activity reported (Amiram Grinvald and Hildesheim 2004). The very fast response times of VSD may make it more difficult to use, but makes it ideal for neuroimaging of spontaneous, resting-state infraslow ($<$0.1Hz) brain activity as then VSD imaging directly reflects underlying electrical activity with high spatial and temporal resolution allowing for imaging of nearly the entire dorsal neocortex while maintaining high spatial resolution and at rates sufficient

to resolve multiple time signatures of ongoing activity (Chan et al. 2015). The dendrites of cortical cells are often far more confined than the axons, so most of the signals in a given pixel originate from the dendrites of nearby cortical cells. VSD signal *in vivo* mainly reflects dendritic activity and as dendrites of one cell may overlap with another VSD activity in one region may not mean cortical activity in that region is being elicited (Malach et al. 1993). This brain activity localization issue can however be partially offset with image-processing approaches offering better spatial resolution such as differential imaging (Blasdel and Salama 1986). Although pharmacological side-effects including phototoxicity have been minimized in newer dyes (Shoham et al. 1999) VSD remains ill-suited for longitudinal imaging of spontaneous activity in behaving animals as eventually the dye needs to be reapplied as voltage sensitivity wanes. Long-term (≤ 1 year) repeated VSD application and imaging of the same cortical area however can be made to not disrupt normal cortical architecture as has previously been confirmed in monkeys (Slovin et al. 2002). Devising an experimental setup to bypass VSD's potential phototoxicity has a cost. The intrinsic noise in VSD imaging of the awake monkey was larger than observed if the monkey was anaesthetized (Slovin et al. 2002). Anaesthesia has long been used in living animals to allow for the imaging of processes longitudinally and noninvasively. Anaesthetic agents however can have unintended effects on animal physiology that may introduce confounds. In addition, repeated anaesthesia, animal preparation for imaging, exposure to ionizing radiation and the administration process may all affect the process under study (Hildebrandt, Su, and Weber 2008). Since animals need to be restrained during this lengthy procedure, there is concern that measured activity do not accurately mimic what brain activity would look like were the mouse in its natural environment. So although organic VSDs typically have fast kinetics that may make them invaluable for mapping infraslow spontaneous activity of the functional mesoscale cortical connectome, they are often phototoxic, allow neither genetically targeted delivery nor long-term imaging studies of single cells, and have been incapable of reporting single spikes in the live mammalian brain (Peterka, Takahashi, and Yuste 2011; Michael Z. Lin and Schnitzer 2016). These are limitations that are generally shared by other organic dye methods and as such it is expected that the advantages of genetically encoded indicators will surpass the traditional use of organic dyes for many applications and, in particular, for circuit analysis of brain functions (Knöpfel 2012). Fura-2, a ratiometric calcium indicator organic dye (Grynkiewicz, Poenie, and Tsien 1985) has for instance been largely replaced by GCaMP6 that simply offers a far stronger signal. As such only VSD, which sees use by our lab, is covered here.

1.2.2 Intrinsic Signal Imaging

For functional imaging of spontaneous brain activity, including blood dynamics, two techniques are usually employed.

When a photon scatters from a moving particle (e.g. hemoglobin in blood), its carrier frequency is Doppler shifted, scattering the light and producing an interference pattern called a speckle which can be used measured to infer the particle's movement (e.g. blood flow) (Amiram Grinvald, Sharon, et al. 2016). In neuroimaging, Laser Speckle Imaging (LSI) is broadly categorized into using either temporal or spatial contrast (Boas and A. K. Dunn 2010; Basak, Manjunatha, and Dutta 2012). Temporal contrast offer better spatial resolution at the expense of temporal resolution, whereas spatial contrast offers better temporal reso-

lution at the expense of spatial resolution (Boas and A. K. Dunn 2010; Basak, Manjunatha, and Dutta 2012). This trade-off is an intrinsic limitation.

Changes in optical diffuse reflectance of the exposed cortex attributed to local changes in cortical blood volume and oxygenation of the brain are visible, even without application of dyes (i.e. intrinsic signals) (A. Grinvald, Lieke, et al. 1986; Y. Ma et al. 2016; Malonek and A. Grinvald 1996; Rector et al. 2005). Intrinsic Optical Signal (IOS)s originate from changes in several intrinsic sources of brain activity including: blood flow and oxygen consumption affecting hemoglobin saturation level, changes in blood volume affecting tissue light absorption, changes in light scattering due to ion and water movement in and out of neurons, changes in the volume of neuronal cell bodies, capillary expansion, and neurotransmitter release (Ron D. Frostig and Chen-Bee 2009; A. Grinvald, R. D. Frostig, et al. 1988). The relative contribution of these various sources to the IOS is dependent on the wavelength of light used to illuminate the cortex. (Ron D. Frostig and Chen-Bee 2009). Blood flow is best obtained using blue light illumination whereas cortical activity combined with blood oxygenation is obtained using red or orange light (R. D. Frostig et al. 1990). Unlike VSD, IOS and LSI are are label-free and therefore suffers from no setbacks when used in longitudinal studies (Vanni and Timothy H. Murphy 2014; Amiram Grinvald, Sharon, et al. 2016). However, IOS sensitivity is lower than other indicators and so is its accuracy in observing remote activity in connected regions during spontaneous activity (Vanni and Timothy H. Murphy 2014). Local activations evoked by both sensory and cortical microstimulation are more than one order of magnitude weaker when measured using IOS versus GCaMP3 (Vanni and Timothy H. Murphy 2014). Although the signal-noise-ratio for GCaMP3 and IOS were similar within regions of peak activity, long-range connectivity patterns of IOS display less consistent and weaker patterns with more areas of patchy noise (Vanni and Timothy H. Murphy 2014).

Finally, intrinsic signals often measure more variables than is practical. This is in contrast to genetically encoded indicators that are engineered to indicate specific activity. Consequently, the neurophysiological interpretation of intrinsic signals, such as from IOS or LSI, are not as well understood and the calibration often far more complex. Further research is still required to determine the exact correspondence between intrinsic signals and AP's, local field potentials, and glial sources (Ron D. Frostig and Chen-Bee 2009).

1.2.3 Genetically Encoded Indicators

Genetically Encoded Indicator (GEI)s can be stably expressed to study how vesicle release, changes in neurotransmitter concentrations, transmembrane voltage, and intracellular calcium dynamics evolve over time in individual animals during the course of learning, life experience, brain development, or disease progression (Dana et al. 2016; Peters, S. Chen, and Komiyama 2014; Ziv et al. 2013; Michael Z. Lin and Schnitzer 2016). As indicators that are part of an animal's biology from birth to death they are ideal for longitudinal studies requiring high-throughput data collection of spontaneous activity in awake behaving animals. There are many animal preparations that allow minimally invasive optical access or placement of optical probes up to millimetres away from the imaged cells. In mammals, methods such as thinned-skull, cranial-window, and microendoscope preparations allow the immediate vicinity of cells under study to be left unperturbed (Betley et al. 2015; Ghosh et al. 2011), while certain model organisms such as zebrafish are translucent and can be imaged intact. This avoids of local perturbation and typically makes GEIs the

indicators of choice for long-term non-invasive imaging. Achieving the appropriate and desired GEI, it must be noted, can be time-consuming and costly. GEIs are expressed by viral infection or transgenesis via utero electroporation, both which require empirical optimization to test for transduction efficiency, appropriate yield specifically in appropriate cell types and whether it causes toxicity in the cells of interest (Michael Z. Lin and Schnitzer 2016).

Vesicle Release Indicators

The pH of a neurotransmitter vesicle is typically ~5.5, whereas that of the extracellular environment is 7.0–7.5 (Kavalali and Jorgensen 2014). Thus upon neurotransmitter release via membrane fusion, a pH change occurs. Genetically Encoded pH Indicator (GEPI)s are pH-dependent fluorescent proteins, of which Super Ecliptic pHluorin (SEP) has the most complete deprotonation at neutral pH. This is a desired characteristic as the increase in brightness from pH 5.5 to 7.4 is therefore the most pronounced for SEP, allowing for better precision in assessing how many vesicles have released their contents given a particular brightness. *In vivo*, autofluorescence and scattering reduces $\Delta F/F_0$ while decreasing signal photons acquired (Wilt, Fitzgerald, and Schnitzer 2013; Michael Z. Lin and Schnitzer 2016). Re-cycling terminals represent a relatively small fluorescent reservoir relative to other contributors to signal. Therefore, GEPIs are not likely to be useful for imaging single action potentials in vivo and therefore not likely to be useful for single cell imaging for mesoscale mapping of spontaneous brain activity. However, at the whole-brain scale, SEP can visualize responses integrated over large numbers of synapses with observed $\Delta F/F_0$ as high as 100%. This is due to multiple SEP-labelled synapses being present within each imaged pixel. Obtaining this integrated signal in one pixel is possible since vesicular proteins remain at the "higher-pH synaptic cleft" after vesicle fusion for as long as ~1sec (Michael Z. Lin and Schnitzer 2016). Therefore, with long exposure times, multiple fused vesicular proteins fluoresce (due to attached SEP) and the signals are aggregated in a single pixel. With this advantage, SEP was the first GEI used in vivo to visualize population-level neuronal activity in the fly (Ng et al. 2002) and mouse (Tabares et al. 2007) olfactory glomeruli. Despite this, GEPIs can nonetheless often be replaced by GECIs to localize vesicle exocytosis and detect action potentials due to the amplified nature of calcium signals (calcium is used to transport vesicles to the release sites). Orange-red GEPIs capable of improving action potential detection in a single neuron are currently being engineered and refined from pHTomato, pHoran4, and pHuji (Shen et al. 2014; Michael Z. Lin and Schnitzer 2016; Hamel et al. 2015).

Neurotransmitter Indicators

GETIs can be expressed on either pre or postsynaptic cells, allowing visualization of neurotransmission from specific presynaptic or to specific postsynaptic cell types (Liang, Broussard, and Tian 2015). To date the most responsive glutamate GETI is iGluSnFR. The sensor's baseline brightness is far brighter than that of SEP and all known GETIs. Comparisons in cell culture further showed iGluSnFR to be more sensitive than SEP in detecting synaptic release (Hikima, Garcia-Munoz, and Arbuthnott 2016). Like SEP, iGluSnFR has also been successfully used to produce a low-resolution map of activity in the entire mouse brain (Xie et al. 2016). Unlike SEP however, iGluSnFR is bright enough to detect release events attributed to individ-

ual action potentials (Marvin et al. 2013) and unlike VSD, iGluSnFR provides direct measures of synaptic activity (Xie et al. 2016). In sum, iGluSnFR is readily suitable to be applied in *in vivo* preparations and works robustly for long-term imaging with high sensitivity and fast kinetics in various species (Marvin et al. 2013). Mesoscopic cortical imaging of iGluSnFR enables mesoscale mapping of synaptic activity and cortical circuits in awake behaving animals during longitudinal studies of pathological brain states that require higher temporal resolution (Xie et al. 2016). In contrast, GECIs are limited in their use for multi-cell imaging applications to detection of spike-related signals at low frequencies due to their slow kinetics (T.-W. Chen et al. 2013). iGluSnFR has few shortcomings for mesoscale mapping of spontaneous brain activity. Single-vesicle detection remains to be unambiguously demonstrated. However such a scale is not required for mesoscale mapping. Perhaps surprisingly, glutamate is the only neurotransmitter for which there exists a well-characterized GETI (new GETIs are being developed for GABA). Given how the interaction between different neurotransmitters have a range of effects on postsynaptic neuronal activity GETIs for other neurotransmitters will play a key role in dissecting neuronal circuitry. Given the existence of bacterial proteins for acetylcholine, GABA, and glycine (Berntsson et al. 2010) GETIs will likely be engineered for these neurotransmitters eventually (Michael Z. Lin and Schnitzer 2016; Q.-T. Nguyen et al. 2010).

Voltage Indicators

Genetically Encoded Voltage Indicator (GEVI)s show the largest variety of designs and mechanisms (See Fig. 1.1) (St-Pierre, Chavarha, and Michael Z Lin 2015; Knöpfel, Gallero-Salas, and Song 2015). VSFP2- and Butterfly-family GEVIs contain a second fluorescent protein to serve as a Forster Resonance Energy Transfer (FRET) acceptor. An increase in voltage causes a conformational change and both fluorescent proteins emit different wavelengths with one increasing in brightness and the other decreasing. Dividing the fluorescence of one by the other results in a ratiometric measure of voltage activity (Knöpfel, Gallero-Salas, and Song 2015). FlicR1 has a single fluorescent protein domain attached to the voltage sensing domain C terminus and, unlike other GEVIs, brightens upon depolarization instead of dimming (Abdelfattah et al. 2016). This diversity is perhaps because no single GEVI design has yet met all the desired performance requirements satisfactorily. GEVIs would ideally have fast millisecond-scale kinetics, reliable membrane targeting to specific cell types and enough brightness and suitable responsiveness to discern single-cell action potentials and subthreshold voltage activity. VSFP-Butterfly1.2, and Mermaid2 feature fast activation kinetics ($<$3ms) and evolved responses of single cells were detectable by averaging 10s trials (Akemann et al. 2013). However, inactivation kinetics remain slow ($>$10ms) (Michael Z. Lin and Schnitzer 2016). This can however be useful when detection of APs and subthreshold voltage changes is desired by sub-millisecond timing is not necessary (Michael Z. Lin and Schnitzer 2016). Arclight can be used to detect action potentials and subthreshold depolarizations in single cells in flies (Cao et al. 2013), but distinguishing between the depolarization and hyperpolarization is difficult because of the slow kinetics of ArcLight for both activation and inactivation (10 ms and 28 ms respectively) (St-Pierre, Marshall, et al. 2014; Michael Z. Lin and Schnitzer 2016). Even when considering the signals created by sensors that have both fast activation and fast inactivation (ASAP1, Mac-Citrine, and Ace-mNeonGree) there remain issues. These GEVIs suffer from signals that do not persist appreciably beyond the 2-ms durations of action potentials. Very

fast (>300Hz) sampling is thus required to detect action potentials using these indicators. Such sampling rates in turn require high excitation intensities to achieve reasonable rates of event detection, causing faster bleaching which limits experiment duration (T.-W. Chen et al. 2013; Michael Z. Lin and Schnitzer 2016; Zou et al. 2014). As no existing GEVI combines all desirable features, GEVI engineering continues to be an active area of research.

Calcium Indicators

Calcium imaging allow for the study of neural ensemble dynamics and coding (Ziv et al. 2013; Ahrens, Orger, et al. 2013; Dombeck et al. 2010; O'Connor et al. 2010), dendritic processing (Sheffield and Dombeck 2015), synaptic function (Michael Z. Lin and Schnitzer 2016), and chronic preparations allowing long-term time-lapse imaging (Dana et al. 2016; Peters, S. Chen, and Komiyama 2014; Ziv et al. 2013). Calcium imaging is arguably the most mature modality for optical imaging of neural activity (Michael Z. Lin and Schnitzer 2016). The brightest GECI family, GCaMPs, are generated from a fusion of green fluorescent protein, calmodulin, and M13, a peptide sequence from myosin light chain kinase (Akerboom, Rivera, et al. 2009; Ding et al. 2014). Major limitations of GECI imaging are that they have relatively slow kinetics, limiting their use in detecting multi-cell spike-related signals at low frequencies, neurotransmitter receptor activation or action potential firing with temporal precision due. GECIs also do not report membrane hyperpolarizations (Gore, Soden, and Zweifel 2014) or subthreshold voltage changes well (T.-W. Chen et al. 2013). Finally, although GECI signals can be large, they can non-linearly relate to calcium concentration (Rose et al. 2014), which in turn exacerbates analysis of the relationship between calcium activity and membrane voltage (Berger et al. 2007). Consequently, algorithms that calculate spike rates from GECI data only achieve 40–60% accuracy for higher frequency events (Theis et al. 2016). Single-fluorophore GECIs have achieved larger responses than FRET-based GECIs (YC2.60, YC3.60, D3cpv, TnXL81 and the Twitch-family, See Fig. 1.1). Currently the most commonly used GECIs are GCaMP-family GECIs (see See Fig. 1.1) and of this family the most used are those of the GCaMP6 series. GCaMP-family GECIs have since been improved iteratively by several groups through many rounds of mutagenesis and selection (T.-W. Chen et al. 2013). GCaMP6f reports single APs in mouse cortex with 20% $\Delta F/F_0$, superior to that of organic dyes (T.-W. Chen et al. 2013). GCaMP6m, GCaMP6s, and GCaMP7 produce even larger responses to single action potentials, but at the cost of decays that are 93–190% longer than that of GCaMP6f (T.-W. Chen et al. 2013; Podor et al. 2015). A variant of GCaMP3, GCaMP3fast has fast kinetics with an in vitro half-decay time of 3 ms, and has lower baseline fluorescence, however its performance *in vivo* has yet to be characterized (Helassa et al. 2015). GCaMPs therefore remain useful due to their variant properties. GCaMPs have since been used to visualize activity in entire worm and fish brains, implicating specific neurons in decision-making or learning (Ahrens, J. M. Li, et al. 2012; T. W. Dunn et al. 2016; J. P. Nguyen et al. 2016; Venkatachalam et al. 2016). GCaMP has been used to map functional circuits activated due to learning in the mouse (Peters, S. Chen, and Komiyama 2014; Ziv et al. 2013; Lovett-Barron et al. 2014). GCaMPs have also been used to localize activity to specific postsynaptic and presynaptic compartments (Siegel and Lohmann 2013). For example, GCaMP6s was used to identify how visual stimuli at different orientations elicit different responses at different spines or axonal boutons within the same visual cortex neuron (T.-W.

Chen et al. 2013; Sun et al. 2016). Finally, GECIs can be used instead of GEPIs to localize synaptic vesicle exocytosis due to the amplified nature of calcium signals (calcium is used to transport vesicles to the release sites). GECIs can therefore replace GEPIs in some experimental setups. An exciting current development is the production and refinement photoconvertible GECIs that change wavelength upon illumination. This allows optical selection of neurons for activity tracking or optical marking of neurons that exhibit interesting activity patterns for later follow-up (Michael Z. Lin and Schnitzer 2016). For example, the unique GECI CAMPARI undergoes green-to-red photoconversion by 400-nm light allowing it to permanently mark neurons activated by specific behaviour or sensory stimulation (Fosque et al. 2015).

1.2.4 Conclusion

Taken together, many optical methods exist to effectively observe different aspects of mesoscale brain dynamics of spontaneous brain activity. Although varied and each with its own pros and cons it must be noted that the processing pipeline that image stacks go through from image acquisition to visualizations that can be used to interpret data is largely the same, with some exceptions, across all optical imaging modalities. Image stacks are trimmed of obvious extrinsic noise, hemispheric or regions-of-interest are cropped, image stacks are aligned to a common coordinate axis, temporal and spatial filtering is applied and $\Delta F/F_0$ is computed and often plotted for regions-of-interest. Of course different file formats might be used, different filters, different baselines for $\Delta F/F_0$ and different coordinate axis scales and alignment protocols but the same general algorithm is at place. Exceptions include SEP that might also require a processing step aggregating many signals into a single pixel with long exposure times. Ratiometric indicators such as VSFP2- and Butterfly-family GEVIs require that two wavelengths are recorded and that a division step occurs between the activity in both recorded at each wavelength. However, after this step all other processing steps are similar. Finally, new upcoming photoconvertible GECIs may require additional processing steps from image acquisition before being interpreted. All in all, this suggests that as long as researchers are aware of the limitations of each aforementioned imaging technique and data is properly acquired that a common computational toolbox can then be shared by neurophotonic researchers. Such a toolbox would be method agnostic and broadly usable, but also specialized and optimized specifically for optical imaging of spontaneous activity in awake animals for mesoscale functional connectivity analysis.

1.3 High-throughput Functional Mesoscale Mapping

Longitudinal assessment of cortical activity through animal head-fixed behaviour coupled with functional imaging requires significant investigator involvement. Consequently, our lab has recently developed a system for high-throughput automated head-fixing and mesoscopic functional imaging of transgenic mice (Timothy H. Murphy et al. 2016). Similar methods have been developed for rats (B. B. Scott, Brody, and Tank 2013). Transgenic mice are head-fixed automatically and their brain activity is imaged through a bilateral transcranial window encompassing much of the cortex. The self-contained nature of our system means mice are permitted to remain in an animal facility preserving colony integrity and facilitating high-throughput cortical physiology under researcher-defined conditions.

The increasingly high-throughput nature of functional mesoscale data collection increases the amount of

data to be assessed, necessitating the need for automated processing pipelines. This trend is analogous to the increasingly high-throughput nature of structural mesoscale mapping where computer algorithms are relied on to automatically examine each section in the topography and, importantly, to standardize approaches including pipeline processing and how data is visualized after processing. This is true even in the case of the Rat Neurome Project (Swanson, Sporns, and Hahn 2016; Bota, Sporns, and Swanson 2015) where computer algorithms are used to efficiently store rat connectomic data from previously published literature in a single standardized format.

Be this as it may, a second trend to be noted in high-throughput structural mesoscale mapping projects is that, despite technological innovations to improve efficiency, for the sake of quality control direct human analysis remains a bottleneck. University of Southern California neuroscientist Houri Hintiryan, who is working to generate a mouse mesoconnectome using multiple coloured tracers (Zingg et al. 2014), says that the gold-standard tool for this analysis is still the human eye. Lining up structural and sequential images still has to be done partially by a person, rather than being fully automated, which can be physically exhausting (Dance 2015). The microscale FlyEM project at the Howard Hughes Medical Institute's Janelia Research Campus aims to map behaviourally relevant circuits in the entire drosophila connectome (Dance 2015). However, despite technological advancements and automation in repeating the scanning of brain slices over 500,000 times per brain, Janelia scientists still do not fully trust the computer yet. Accuracy is noted to be at 95% for how well their algorithm identifies cells and synapses, which is far too low to not explicitly require human proofreaders (Dance 2015). Humans still do data processing best in many important stages of the processing pipeline required to map an organism's brain.

This is not to say that these challenges cannot be automated away with more robust algorithms. Partha Mitra of Cold Springs Harbor Laboratory in New York is working on building a robust virtual neuroanatomist that would make sense of mesoscale imaging slides, thereby automating the pipeline of mapping the mouse structural mesoscale connectome (Dance 2015). Similarly, Jeff Lichtman of Harvard University in Cambridge and his colleagues are working on an algorithm to automate cell and synapse identification with higher accuracy (Dance 2015).

However, in the interim - before such automation solutions bear fruit from experts working to develop them - better software is required that make it easier on the user to perform the analysis on parts of the processing pipeline of high-throughput mesoscale functional mapping that cannot be reliably automated.

Researchers such as Helmstaedter and Seung have realized the need for an interim solution for structural mapping and have opted for crowdsourcing the challenge to lay-people. Helmstaedter developed a game, called Brainflight, in which players 'fly' through the brain's nerves and software captures those movements to the define the borders of the axons (Helmstaedter 2013) Likewise, Seung's group has developed the human based computation game project EyeWire in an attempt to map the connectome of the retina (Tinati, Luczak-Roesch, Simperl, and Hall 2016; Tinati, Luczak-Roesch, Simperl, Shadbolt, et al. 2015; Helmstaedter 2013). Although both examples deal with creating experiences that are entertaining so as to attract lay-people - an aspect of software development not at all explored in this thesis - it is instructional to note that as interim solutions, while more robust automation algorithms are developed, these projects do capitalize on software that is easily usable by users inexperienced in programming. Both interim solutions can also be modified by

users at the graduate or undergraduate level in attempts to enhance the interface or entertainment value.

Both the aforementioned interim projects are for microscale analysis. In mesoscale functional mapping of the mouse brain no feasible alternative solution exists to the bottleneck of human proofreading required to assure quality in high-throughput processing pipelines. Despite this, very few interim software solutions exists to standardize approaches with an easy to use interface to make manual analysis easier for researchers while such proofreading remains an unfortunate bottleneck in mesoscale functional mapping processing pipelines (Haupt et al. 2017).

1.4 Research Aims

We implemented Mesoscale Brain Explorer (MBE), a flexible open-source Python-based image analysis and visualization tool. Our goal is to implement user-friendly software to standardize and automate common processing steps and analysis in mesoscale wide-field optical mapping to enhance reproducibility of mesoscale brain mapping pipelines. Common processing steps are those that are found across multiple optical imaging modalities (See Sect. 1.2.4). See Table 2.1 for a list of common processing steps we have identified and implemented. At a minimum we require the application to proceed from raw data and correctly process data through user-selected processing steps available in the application.

We also require that output visualizations commonly used in our lab to explore spontaneous mouse brain activity from VSD or GEI imaging data be supported in the application for easy data exploration. For the purposes of this thesis we focus on ensuring MBE can correctly output brain activity plots, correlation matrices and seed-pixel correlation maps. These visualizations are among the technically simplest to implement with the easiest neurophysiological interpretation upon output. See Sect. 2.5 for results and neurophysiological interpretations garnered from these visualizations. We therefore chose to focus on these so that the focus can instead be on the design and correctness of the application. Visualization output must be correct based on user-selected parameters for the processing steps - the interplay of which can be as complex as the user decides (See Sect. 4.1.1 for an example) - taken to reach the output.

Support for further far more complex visualizations might be added later. See Sect. 4.2.2 for a discussion of future possible visualizations. Therefore, principally for this thesis, the goal is for MBE to act as a framework wherein further processing or visualization tools can later be added to it.

To Summarize, our goal in implementing MBE is to create an application that is method agnostic and thus broadly usable, but also specialized and optimized specifically for optical imaging of spontaneous activity in awake animals for mesoscale functional connectivity analysis. To achieve such a goal effectively we have two aims for MBE

- Aim 1: A neurophotonics researcher inexperienced with programming can easily manage and share their own processing pipeline within the application
- Aim 2: MBE is easy for programmers at the undergraduate level to modify and extend their own functionality onto without having to rewrite modules used often in the program

Aim 1 deals with MBE as an effective exploratory tool for mesoscale connectivity mapping and is covered in Chapter 2 which can be thought of as the "Results" chapter. Aim 2 deals with what happens "under the

hood" and principally is concerned with making it as easy as possible to get someone to modify MBE for their own purposes. Aim 2 is covered in Chapter 3 and can be thought of as the "Methods" chapter.

Chapter 2

Mesoscale Brain Explorer

2.1 Introduction

Among the initial goals of the brain initiative was to map the functional activity of potentially every neuron within the human brain (Devor et al. 2013). While this challenge has led to many new approaches to assess connectivity (Silasi and Timothy H. Murphy 2014; C. Liu et al. 2016; Vanvinckenroye et al. 2016; Ajetunmobi et al. 2014; Hagen et al. 2015), it is probably unattainable in the near term. An equally important level of resolution to assess functional relationships is the mesoscale. The mesoscale is an intermediate level of brain functional connectivity between the microscale of cells and synapses and macroscale connections best visualized using whole brain Functional Magnetic Resonance Imaging (FMRI) methods (Devor et al. 2013). Through the pioneering work of Grinvald, his colleagues and others, mesoscale connectivity analysis has been well-established (Petersen et al. 2003; Kenet et al. 2003; Carandini et al. 2015; Kleinfeld and Delaney 1996; Vanni and Timothy H. Murphy 2014; Majid H. Mohajerani, Chan, et al. 2013). Results from feline and rodent cortex nicely demonstrate the role of large ensembles of neurons which contribute to cortical maps that are shaped by experience and are associated with particular behavioural states (Petersen et al. 2003; Kenet et al. 2003). Complementing the classic strategies of Grinvald (Amiram Grinvald and Hildesheim 2004) and Petersen (Ferezou et al. 2007) are more recent structural connectivity analyses performed by the Allen Institute (Harris et al. 2014; Oh et al. 2014) and others (Hunnicutt et al. 2014; Zingg et al. 2014) where the projection anatomy of most mouse brain areas can be mapped into a common coordinate framework for C57BL/6 mice.

The level of resolution afforded by mesoscale imaging provides opportunities to compare data across imaging modalities, species, and behaviours (Tanigawa, Lu, and Roe 2010; Kenet et al. 2003; Slovin et al. 2002; Petersen et al. 2003). Grinvald exploited connectivity relations embedded within spontaneous brain activity in a similar manner to resting state FMRI (Fox and M. Greicius 2010). This analysis performed largely within spontaneous events of the cat primary visual cortex, provided a means of assessing functional connectivity relations, which were also present when the animal was given defined visual stimuli (Kenet et al. 2003). Recently, our lab and others have taken advantage of large field of view imaging within the mouse cortex to also assess functional connectivity using spontaneous activity (Majid H. Mohajerani, Chan, et al.

2013; M. H. Mohajerani, Aminoltejari, and T. H. Murphy 2011; White et al. 2011). This approach, when combined with new structural connectivity information (Harris et al. 2014), indicates that functional connectivity is constrained by major intracortical axonal projections (Lim, Majid H. Mohajerani, et al. 2012; Majid H. Mohajerani, Chan, et al. 2013). This approach of examining relationships within spontaneous events or those stimulated by optogenetics also provides a potential vehicle for broad comparisons between human resting state fMRI studies and the mouse mesoscale connectome. While these advances, facilitated by the careful insight of Grinvald (Kenet et al. 2003), have moved the field forward, a significant hurdle exists in processing and interacting with large data sets of mesoscale functional activity. Accordingly, we have built a flexible open-source Python tool, which permits significant processing of mesoscale imaging raw data and provides a platform with which others can view and interact with archival datasets (such as widefield mesoscale imaging data from transgenic mice) and explore their own regions of interest, frequencies, or other properties. The tool is further designed so that user-specified plugin pipelines can be created to automate processing steps using existing plugin functions or custom plugins with user defined additional functions.

One method for inferring functional connectivity from collected spontaneous data would be through the creation of Seed Pixel Correlation (SPC) maps (Fox, Snyder, et al. 2005): a single pixel (or a small region of interest) is selected as the seed. Pearson correlation (zero lag) is used to generate a map showing the extent to which brain activity over time at each pixel correlates with that of the seed (Vanni and Timothy H. Murphy 2014; Majid H. Mohajerani, Chan, et al. 2013; Majid H. Mohajerani, Chan, et al. 2013).

Correlation matrices are generated from the activity for particular brain Region of Interest (ROI)s across relatively long sequences of spontaneous activity. Each ROI-ROI pair consists of two sets brain activity with a single correlation value for each pair. Pearson correlation coefficients can be computed for each ROI-ROI pair or even all combinations of pixels to generate a connectivity matrix that can be used to infer interareal connectivity relationships (Silasi, Xiao, et al. 2016; White et al. 2011; Majid H. Mohajerani, Chan, et al. 2013; Chan et al. 2015). Using correlation to infer, monitor and quantify connectivity is common practice in experimental research (Majid H. Mohajerani, Chan, et al. 2013; Vanni and Timothy H. Murphy 2014; White et al. 2011). Voxel based (volume pixel) correlation has been used extensively in human research employing FMRI (Fox, Snyder, et al. 2005). In the case of GCaMP6 this would show us how correlated calcium activity is between selected regions over the time period in which the spontaneous data was collected (typically 3-20 min of activity is recorded).

Correlation matrices are forms of functional connectivity analysis. Functional connectivity is defined as the statistical association or dependency among two or more anatomically distinct time-series (Karl J. Friston 1994). Measures of functional connectivity do not provide information regarding causality or directionality (this is further discussed in Sect. 2.6) . If an analysis of how one region influences another is required then experimental changes studied via effective connectivity methods are required which are outside the scope of this paper (Karl J. Friston 1994).

2.2 Materials and methods

2.2.1 MBE executable

MBE is a cross-platform standalone application (at the time of writing pre-release version 0.7.10 is the latest most stable version available under an MIT license from Haupt that can simply be downloaded and run as an executable without having to set up Python or any further dependencies. Moreover, if a Python 3.5 environment has been set up and the user has installed all dependencies (see instructions for setting up dependencies in the README: Haupt 2017) then the program can be run from the main script via an integrated development environment or the command line. This allows the program to be run on platforms that can not run executable files. To date it has been successfully tested on Windows 7, 8.1, 10 and Linux Ubuntu 16.04 systems. Note that a Python 2.7 implementation is not provided as Python 2.7 will reach its end-of-life in 2020. Python 2.7 users are advised by the Python Software Foundation (Aldama n.d.) to port their code to Python 3.5 and we likewise wish to encourage labs to make the switch. A video tutorial that steps through the entire process required to replicate the figures in this paper is provided (See README Haupt 2017). Example image stacks from mouse #0285 used in this manuscript can be downloaded (see README (Haupt 2017)).

MBE takes a plugin approach to data-processing. Each processing step is independently contained. However, plugins and therefore processing steps used in a particular analysis can be selected, ordered and saved via the Configure Pipeline window. See Fig. 2.1b.

MBE imports data in the form of stacked .tiffs or .raws, both common file output formats for many imaging systems. Image stacks in our context refer to xy over time. Data types uint8, float32 and float64 are supported for .raw file imports, while any datatype is supported for .tiffs as long as its data type is specified in the file header and supported by NumPy (Walt, Colbert, and Varoquaux 2011). Multi-channel B&W or RGB .tiffs or .raws may be used, however, only a single channel is imported at a time. A user who wishes to use both red and green channels from a single file has to perform the import routine twice. Thereafter either imported channel data can be operated on in subsequent plugins. All files are converted to Python NumPy arrays (.npy) upon import and all plugins subsequently assume a .npy format. Any image stack file format is compatible with MBE as long as it can be converted to .tiff, .raw or .npy format. In a session all files imported are contained in a single project.

The user is presented with a Graphical User Interface (GUI) window, menu and dialog driven interface elements alongside two panels (Fig. 2.1a). The left panel (red) is used for managing plugins and data common to the project. The right panel (blue) contains UI) elements specific to a selected plugin.

During analysis, each step is performed with intermediate arrays saved to file. The user can process steps one at a time in any order or set up an automated pipeline where output of a prior step is taken as an input to process the next step in the pipeline. Pipeline configuration, file paths, the source stack of a processing step, an origin selected for a particular stack, a list of all manipulations a stack has gone through and its type are all saved to a JavaScript Object Notation (JSON) file in the user-defined project directory. Files can be filtered via a dropdown menu (the topmost dropdown menu in the blue region in Fig. 2.1a) based on what manipulations they have gone through making bulk deletion to save disk space easy. Moreover, as long as

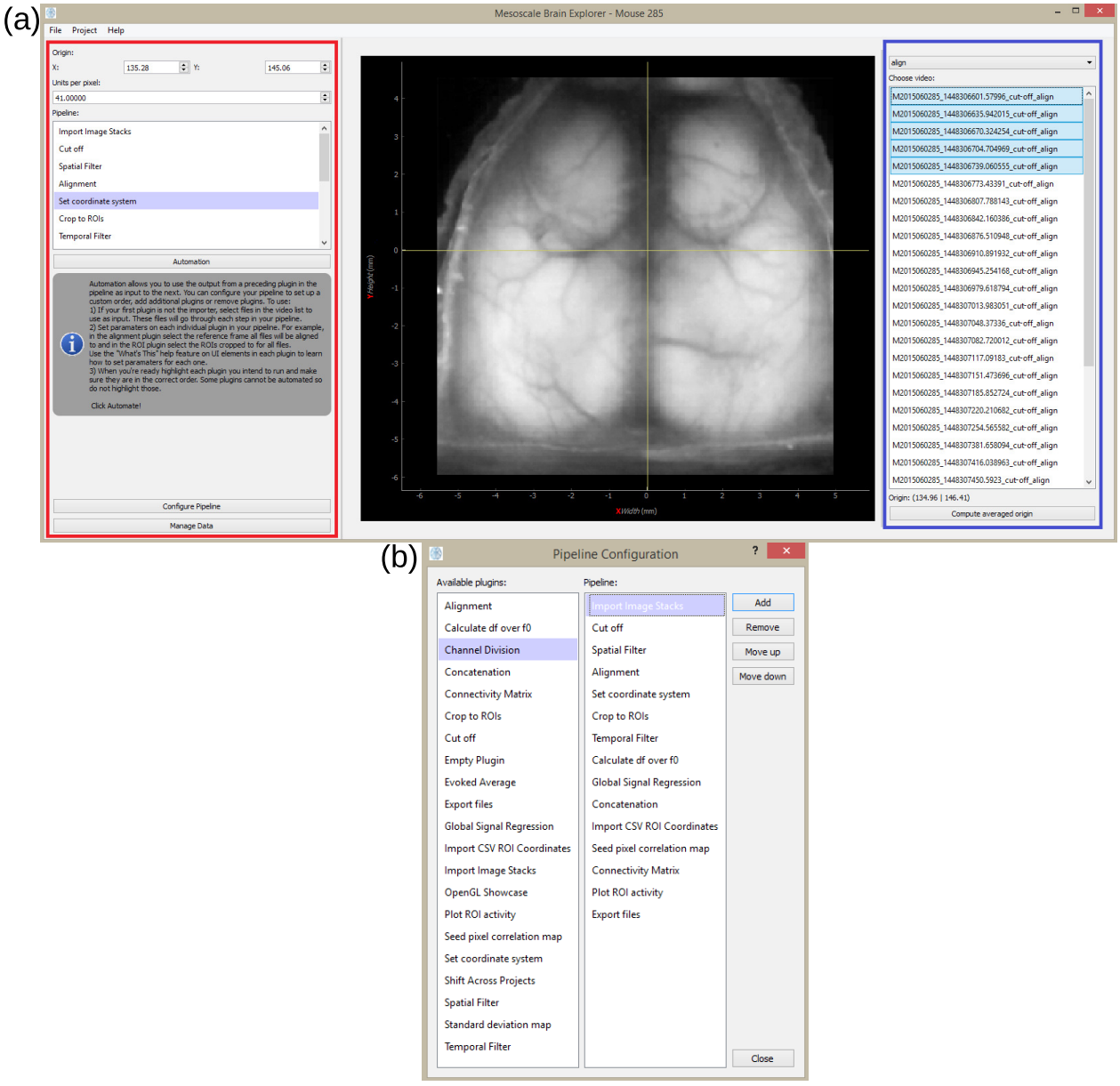


Figure 2.1: (a) The UI includes the left panel (red) for managing plugins and data common to the project such as coordinate system origin and pixel width which here has been set to 41 µm per pixel. The right panel (blue) contains UI elements specific to a selected plugin. Here we have the "set coordinate system" plugin in view. This plugin is used to set the origin and as the pixel width for the project. Here we can see that for this project the five image stacks have been selected. For each one the anatomical location of bregma was clicked and the origin was taken as the average of all five clicks. (b) Plugins and processing steps used in a particular analysis can be selected and ordered via the Pipeline Configuration window

all data and the JSON file are kept together in a single folder with no subfolders, the project can be copied to any supported computer and opened there by MBE with all data and selected processing steps already organized.

MBE is standalone application and does not assume that the user is familiar with Python or the command line. This makes it usable by both programmers and non-programmers. Moreover, source code is structured in a readily extensible framework that can be expanded upon with new plugins developed to suit the specific needs of a researcher (a tutorial on developing your own plugin will be provided in the README). For example, implementing support for different file formats, bandpass filtering techniques or including additional colourmap options for SPC maps (See appendix A.14, A.20 and A.16) are all possible avenues for further development.

2.3 Experimental

Spontaneous activity collected from an awake female Ai94 mouse (Madisen et al. 2015) that was previously published was used in this paper's analysis (Timothy H. Murphy et al. 2016). The mouse was head-fixed automatically whenever the mouse entered a chamber to reach its water spout. Brain activity was subsequently imaged through a bilateral transcranial window encompassing the cortex for 30-64s epochs using a (Wave Share Electronics RPi Camera (F)) Raspberry Pi camera at a frame rate of 30Hz with automatic exposure and auto white balance turned off and white balance gains set to unity. A plastic adjustable lens (f=3.6 mm; provided with the camera) was used after unscrewing the lens and placing a 10-mm-diameter green emission filter (ET525/36m, Chroma Technology) between the lens and the imaging sensor. The use of this camera and lens resulted in a bilateral $10.5 - 10.75 \times 10.5 - 10.75$ MM field of view and imaging occurred through intact bone (Silasi, Xiao, et al. 2016).

Sequences of green epi-fluorescence images using the Raspberry Pi camera are then collected when the mouse is head-fixed. A simple epi-fluorescence system was used with an LED light source (with excitation 475/30m and emission filter ET525/36m Chroma). Collected data (256×256 pixel image stacks) were saved to raw RGB 24-bit files (Timothy H. Murphy et al. 2016). A total of 31 such image stacks were recorded.

All the procedures were conducted with approval from the University of British Columbia Animal Care Committee and in accordance with guidelines set forth by the Canadian Council for Animal Care.

2.4 Theory/Calculation

The pipeline we set up specifically for our analysis of mouse #0285 can be visualized in Fig. 2.2. Note that in the application the ordering of this pipeline can be freely rearranged. Moreover, many additional available plugins (see Appendix A; Table 2.1) can also be inserted anywhere in the pipeline. Many of these we do not use in the analysis covered by this paper.

2.4.1 Initial Preprocessing

The second channel (green) from 31 256x256 pixel raw image stacks was imported into MBE with no resizing (see appendix A.14).

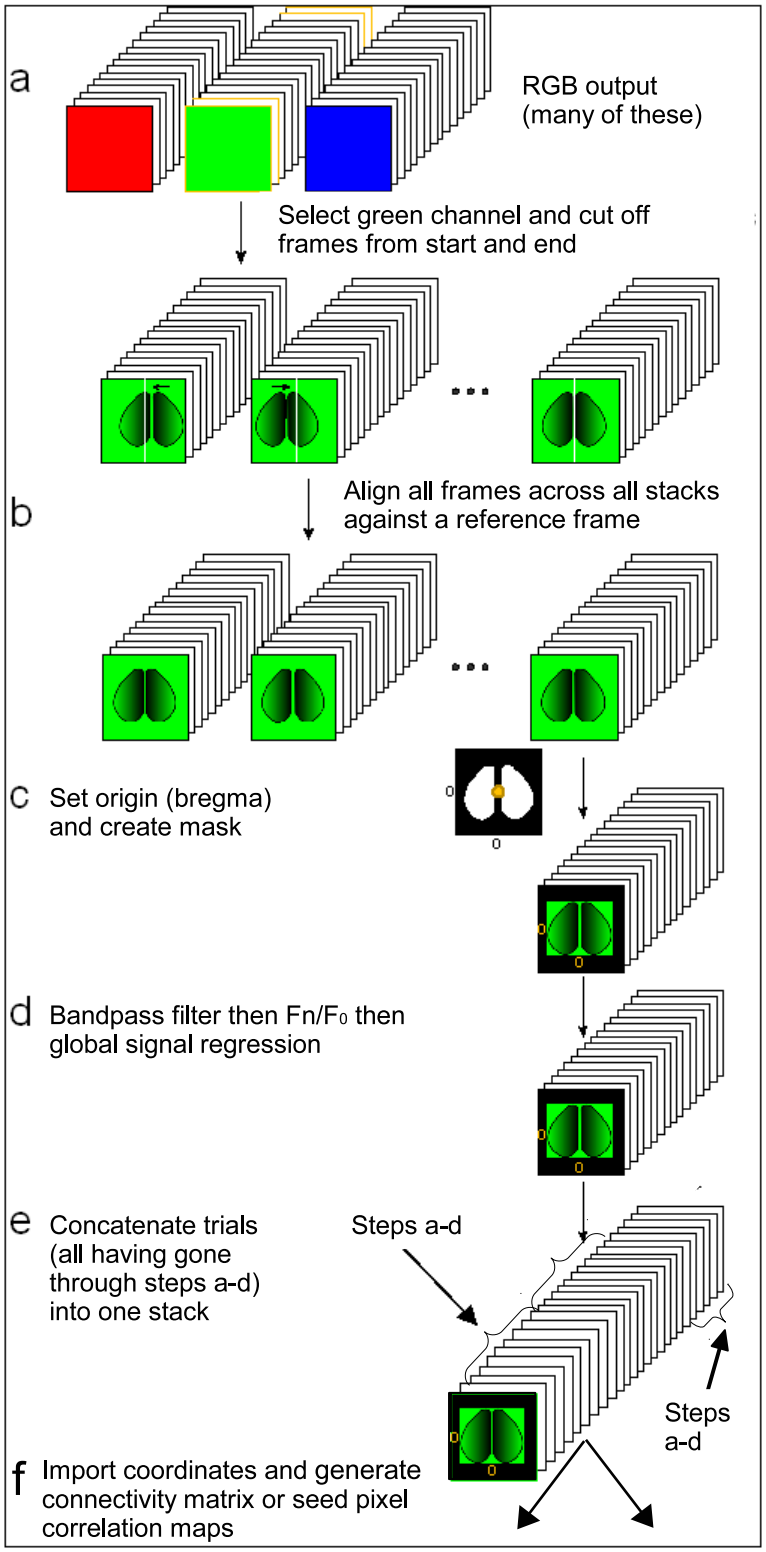


Figure 2.2: The pipeline we set up specifically for our analysis of mouse #0285. In the application the ordering of this pipeline can be freely rearranged.

In our auto-head fixing home cage (Timothy H. Murphy et al. 2016), mechanical settling of the head fixing mechanism results in movement at the beginning of the recording and we therefore delete the first 20 frames in some image stacks as a precaution (Fig. 2.2a).

A single image stack was selected as the template that other stacks were aligned to. All frames in this image stack were sharpened via the unsharp filter plugin (see appendix A.22) using a kernel size of 8. We have previously found (Timothy H. Murphy et al. 2016) through trial and error that this kernel size sharpens each frame to adequately emphasize the location of blood vessels.

The 400th frame ($\pm$13.3s following trim) to the 500th ($\pm$16.6s following trim) of this sharpened image stack were averaged to emphasize the location of the blood vessels further and de-emphasize other features, this step is optional and used to fine-tune alignment. Users may opt to simply select a single frame without performing any averaging. This single averaged frame is set as the reference frame. All frames across all image stacks were aligned to this reference frame and aligned to features that were filtered to produce the reference frame - in this case blood vessels. A fast Fourier transform was used to translate, rotate and scale one user-selected frame from each image stack to align it to the reference frame. The translation, rotation and scale required for this transformation was then applied to all frames in that image stack (Fig. 2.2b). This plugin therefore assumes there was negligible movement within a single image stack (see appendix A.2).

The field-of-view for the recordings is 10.5MM meaning each pixel is 41 µm wide (Fig. 2.1a). The skull anatomical landmark bregma was identified on the first frame of 5 image stacks via the Set Coordinate System plugin. These five locations were averaged to set the origin globally across all plugins (136.28 pixels, 145.06 pixels)(see appendix A.17, Fig. 2.2c). This averaging is done to reduce human error that might occur when clicking the location of bregma.

Polygon ROIs were drawn for both left and right hemispheres, masking the cortex border that was imaged as well as most of the brain midline due to the obstructing midline sinus. These are masked as they are sources of non-neuronal noise. In our example all 31 post-alignment image stacks were cropped to the same ROIs (see appendix A.7, Fig. 2.2c).

2.4.2 Filtering

A Chebyshev filter (Type I digital and analog filter design, order = 4, maximum allowable ripple in pass-band = 0.1) with bandpass of 0.3-3.0Hz was applied to all post-cropped image stacks (see appendix A.20, Fig. 2.2d). This increases the signal-to-noise ratio by removing noise such as cardiac factors (Vanni and Timothy H. Murphy 2014; Carandini et al. 2015). Next, the average across all frames was computed to establish a baseline. The change in fluorescence from this averaged baseline for each frame was computed ($\Delta F/F_0$). This processing step results in data more robust against slow drifting of the baseline signal and fast oscillatory noise due to tissue pulsation, thus ensuring the signal detected more accurately represents brain activity (Jia et al. 2011)(i.e. calcium, glutamate or voltage transients, see appendix A.3). Although available as an option, no image sharpening (i.e. via an unsharp filter) was performed (see appendix A.22) other than to create a reference frame used in alignment (see 2.4.1).

2.4.3 Global Signal Regression

Global Signal Regression (GSR) was applied to all post-$\Delta F/F_0$ image stacks (Fig. 2.2d), except for Fig. 2.4 and Fig. 2.6 where GSR was skipped. GSR is a preprocessing technique for removing spontaneous fluctuations common to the whole brain (Fox, D. Zhang, et al. 2009). GSR involves computing an image stack's global signal that is calculated by averaging the signal across all pixels. The global signal is assumed to reflect a combination of resting-state fluctuations, physiological noise (e.g. respiratory and cardiac noise), and other non-neural noise signals. GSR involves a pixel by pixel removal of the global signal by applying a general linear model. GSR has been shown to remove potential global sources of noise, to heighten the contribution of local networks as opposed to brain-wide transitions and thereby facilitate the detection of localized neuronal signals and improve the specificity of functional connectivity analysis (Fox, D. Zhang, et al. 2009; Timothy H. Murphy et al. 2016; Vanni and Timothy H. Murphy 2014). GSR can also be applied to raw data and this may be advantageous if the image contains areas of variable brightness as low signal areas may be dis-proportionally weighted.

2.4.4 Concatenation

The entire set of all post-GSR 31 image stacks were concatenated and SPC maps computed for all seeds across the concatenated time series (Fig. 2.2e). This is done to use as much spontaneous activity data as possible to improve SPC map accuracy.

2.4.5 Seed Placement

We have previously mapped functional and anatomical coordinates of transgenic mice, confirmed using sensory stimulation in combination with in vivo large-scale cortical mapping using CHR2 stimulation (Lim, Majid H. Mohajerani, et al. 2012). A Comma Seperated Value (CSV) file was made with coordinates in microns relative to bregma for Anterior Cingulate (AC), Visual Cortex (V1), Secondary Motor Cortex (M2), Barrel Cortex (BC), Retrosplenial Cortex (RS), Primary Motor Cortex (M1) and the Hindlimb Cortex (HL) for each hemisphere (see appendix table A.1). Coordinates were added to the project via the "Import CSV ROI Coordinates" plugin displayed in Fig. 2.3b and used as relative distances with respect to bregma (see appendix A.13, Fig. 2.2f). This plugin uses the imported coordinates to create square ROIs of user-specified width centred at those coordinates. The size of the ROIs used for SPC mapping was set to 1. SPC maps were thus computed using single-pixel seeds.

2.4.6 SPC Map and Correlation matrix

SPC maps were generated for all seeds using the concatenated data (Fig. 2.2f). A correlation matrix was constructed from single pixel ROIs using the same coordinates and the 31 post-GSR image stack data.

2.5 Results

2.5.1 SPC Map Generation

Spontaneous activity was collected during the extended head-fixation of a transgenic mouse expressing GCaMP6 (GCaMP6 mouse), mouse #0285. Correlation maps for seed pixels located in right and left V1, BC, HL, M1 and RS were generated (Fig. 2.3e). Maps with seeds M1 and BC reveal intrahemispheric synchronous activity between sensory barrel cortex and motor cortices as previously observed by others (Ferezou et al. 2007; Majid H. Mohajerani, Chan, et al. 2013; Vanni and Timothy H. Murphy 2014).

MBE can output maps to an interactive window (Fig. 2.3b). Pixel values hovered over by the mouse are displayed at the top of the window. The seed label (X, Y position relative to bregma) can be seen at the end of each window title. Each title additionally contains all processing steps performed. This is useful when outputting numerous plots at the same time from various processing pipelines. All maps are additionally saved as .jpeg files automatically. Here we can see that the barrel cortex pixel hovered over has $r \simeq 0.7$ with the M1 seed (Fig. 2.3b. top-left). The user can also click on a pixel to regenerate the map with the selected pixel as the seed.

From the main user interface (Fig. 2.3a) the user can see the first frame of the processed image stack with selected coordinates overlaid. The plugin used for seed placement is shown in Fig. 2.3a. The user can use the right panel in this plugin to load a CSV file that contains micron coordinates. This is displayed in the table in the right panel in Fig. 2.3a. Seeds can be selected via the list in the bottom right. Selected seeds are overlaid and displayed on the brain image in the centre scene where ROIs can be reshaped or moved around. Other plugins will likewise have an interactive scene displaying the first frame of the processed image stack between left and right panels (see Fig. 2.1a).

The image of the first frame between the left and right panel in Fig. 2.3a can be clicked on in the SPC plugin (see appendix A.16) to generate a SPC map for the pixel clicked. The user can additionally select any number of image stacks from the first list in the right panel (identical to the list in the right panel of Fig. 2.3a) and any number of seeds from the list in the bottom of the right panel (identical to the the list at the bottom in the right panel of Fig. 2.3a) to produce SPC maps in bulk for each seed across all selected image stacks.

In Fig. 2.5 timeplots of $\Delta F/F_0$ activity for selected seeds are shown for mouse #0285 with all 31 image stacks concatenated. Only frames (post cut-off) 2100 - 3000 and 2300 - 2700 are shown. In the application the output graph is interactive allowing the user to zoom in on the graph to obtain a clear view of the synchrony between M1 and BC (blue and green), while V1 (orange) is poorly synchronized (This is made more clear in Fig. 2.5c). This is in line with the negative correlation values seen in Fig. 2.3e between V1 and BC or M1.

Pearson correlation coefficients for the full time course of 30604 frames are r_{M1-BC} = 0.685, r_{V1-BC} = -0.397, r_{M1-V1} = -0.522 which agree with the correlation values between these activities in the respective SPC maps (Fig. 2.3e) at these coordinates. All coefficients also agree with r-values previously reported by Silasi, Xiao, et al. (2016): r_{M1-BC} = 0.69, r_{BC-V1} = -0.3, r_{M1-V1} = -0.53. Given the large number of samples all comparisons of BC and M1 activity (with or without GSR) indicated high statistical significance

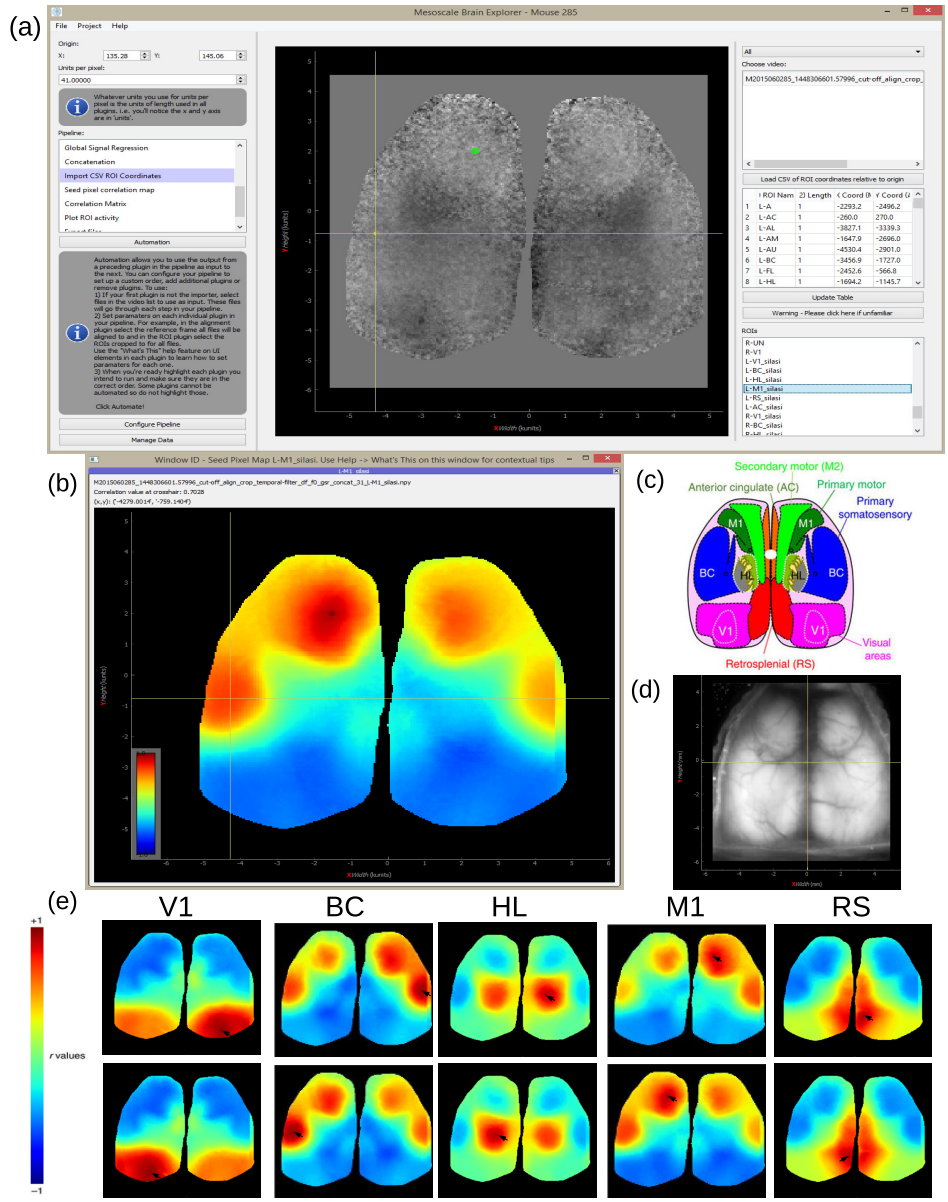


Figure 2.3: SPC mapping for selected seeds with GSR. (a) User interface of the "import CSV ROI coordinates" plugin with M1 seed selected (green ROI) and cross-hair hovering over BC. X and y coordinates for each seed are loaded from a user-defined CSV that is displayed in the table of the right panel. (b) MBE user interface output of SPC Map from the M1 seed in the left hemisphere. The position of seeds for BC and M1 were adjusted to maximize the remote correlation between them. Their positions are still within the general region of motor and barrel cortex (Oh et al. 2014). The correlation value at the cross-hair (BC) is displayed in the top-left of the window (r = 0.7028 with the M1 seed). (c) Atlas of the dorsal region of the cortex (adapted from the Allen Mouse Brain Connectivity Atlas (Oh et al. 2014). (d) Raw green fluorescence data from a single frame from an image stack and the location selected for the skull anatomical landmark bregma that is the origin for the coordinate system. (e) Correlation maps for seed pixel located in right (upper maps) and left (lower maps) V1, BC, HL, M1 and RS.

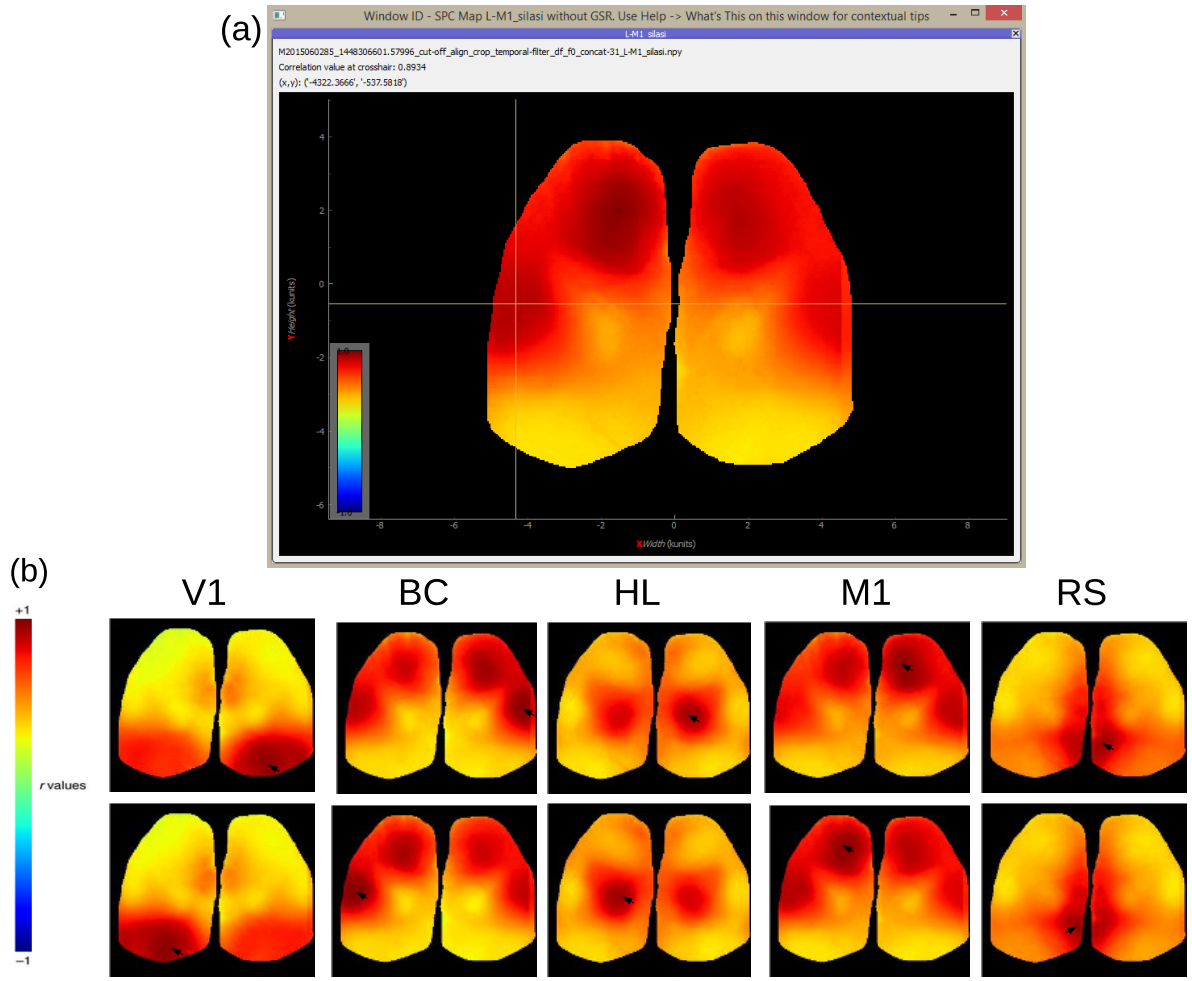


Figure 2.4: SPC mapping for selected seeds without GSR. (a) MBE user interface output of SPC Map from the M1 seed in the left hemisphere. The map is of 31 concatenated $\Delta F/F_0$ image stacks without GSR applied (i.e. GSR was skipped in the pipeline in Fig. 2.2). The position of seeds for BC and M1 were adjusted to maximize the remote correlation between them. Their positions are still within the general region of motor and barrel cortex (*Allen Mouse Brain Connectivity Atlas* 2012). The correlation value at the cross-hair (BC) is displayed in the top-left of the window (r = 0.8934 with the M1 seed). (b) Correlation maps without GSR applied for seed pixel located in right (upper maps) and left (lower maps) V1, BC, HL, M1 and RS.

with p-values $< 1.0 \times 10^{-30}$. We used the barycenter of different regions estimated from Allen Institute anatomical coordinates (*Allen Mouse Brain Connectivity Atlas* 2012). These coordinates do not take into account possible topography of connections which is why the position of seeds for BC and M1 were adjusted to maximize the remote correlation. Coordinates however are still within the general region of motor and barrel cortex (*Allen Mouse Brain Connectivity Atlas* 2012). An advantage of MBE is that the user can open one window for activity plots and another for SPC maps and compare the two to quickly assess the cause of anomalous correlation and adjust coordinates as need be. It is also noteworthy that GSR has been applied to these images to remove global correlations which tends to make all correlations lower (Fox, D. Zhang, et al. 2009; Fox, Snyder, et al. 2005): To compare this with data without GSR see Fig. 2.4 and Fig. 2.6.

2.5.2 Correlation matrix

From the main user interface (Fig. 2.7a) the user can see the first frame of the processed image stack (post $\Delta F/F_0$) with selected ROIs overlaid. The user can select any number of image stacks from the top right list and any number of ROIs (including custom made ROIs that need not be square) to output a single averaged correlation matrix. Correlation matrices are produced for each selected image stack and selected ROIs, but the final output displays correlation coefficients for a single matrix averaged across all matrices. In this example we have selected all 31 post-GSR image stacks from mouse #0285. Pearson correlation zero lag (r-value) was computed for each image stack and for each ROI. These values depict how the ROI correlates with other ROIs in the matrix. Standard deviation of r-values for each ROI-ROI pair is computed, showing the variance of the r-value across image stacks (Fig. 2.7b).

2.6 Discussion

For our analysis, we relied on previously collected recordings of spontaneous activity (Timothy H. Murphy et al. 2016) from awake mice using various fluorescent calcium indicator proteins including GCaMP6 (Madisen et al. 2015; T.-W. Chen et al. 2013). We present an application for visualizing connectivity relationships in these large datasets that makes them more readily available to the scientific community for analysis (our data is available upon request). A limiting issue with studying spontaneous activity is the sheer amount of data that needs to be collected, stored and assessed. Our lab has recently developed a system for high-throughput automated head-fixing and mesoscopic functional imaging for transgenic mice within their home-cages (Timothy H. Murphy et al. 2016). Similar methods were previously developed for rats (B. B. Scott, Brody, and Tank 2013). Consequently, a limiting factor in future longitudinal studies will likely be the ease with which collected data can be processed, analyzed, and shared with the community. MBE was designed to ease processing for the end-user by offering a simple interface and application setup. From a design perspective, a plugin approach was chosen for MBE to enhance usability, maintainability and extensibility. **Usability:** Different processing steps are clearly separated. The program keeps track of data files and pipeline execution, thus the user can focus on their analysis. **Maintainability:** As each processing step resides in its own plugin, functional units are clearly separated from each other and from the base system. This facilitates the understanding of the software architecture and quick localization of faulty code. **Extensible:** Processing steps are added as plugins. Plugins are developed independently from this software.

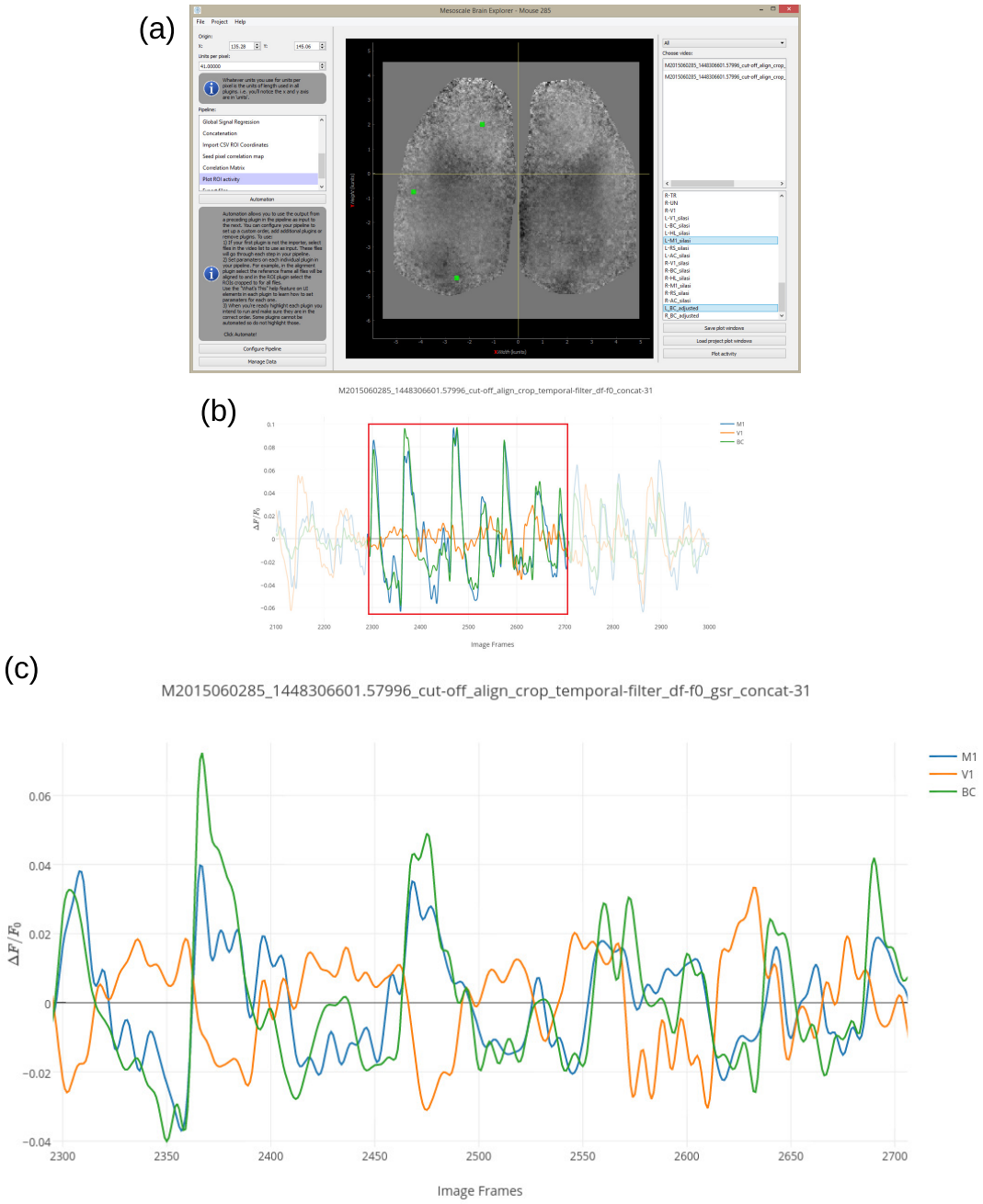


Figure 2.5: Time plots for selected ROI spontaneous activity with GSR. (a) The main user interface with all ROIs to be plotted selected (V1, BC and M1 in the left hemisphere). (b) Zoomed in segment of $\Delta F/F_0$ activity between the 2100th and 3000th frames from all 31 30 second recordings concatenated of spontaneous activity from mouse #0285 (frame rate = 30Hz). r_{M1-BC} = 0.751 p = 1.0253×10^{-164} , r_{V1-BC} = -0.523 p = 1.1418×10^{-64}, r_{M1-V1} = -0.651 p = 8.229×10^{-111}. (c) Further zoomed in segment of $\Delta F/F_0$ activity between the 2300th and 2700th frames highlighting asynchronous activity between V1 (orange) against BC (green) and M1 (blue). Correlation coefficients for the full 30604 frame time course: r_{M1-BC} = 0.685, r_{V1-BC} = -0.397, r_{M1-V1} = -0.522

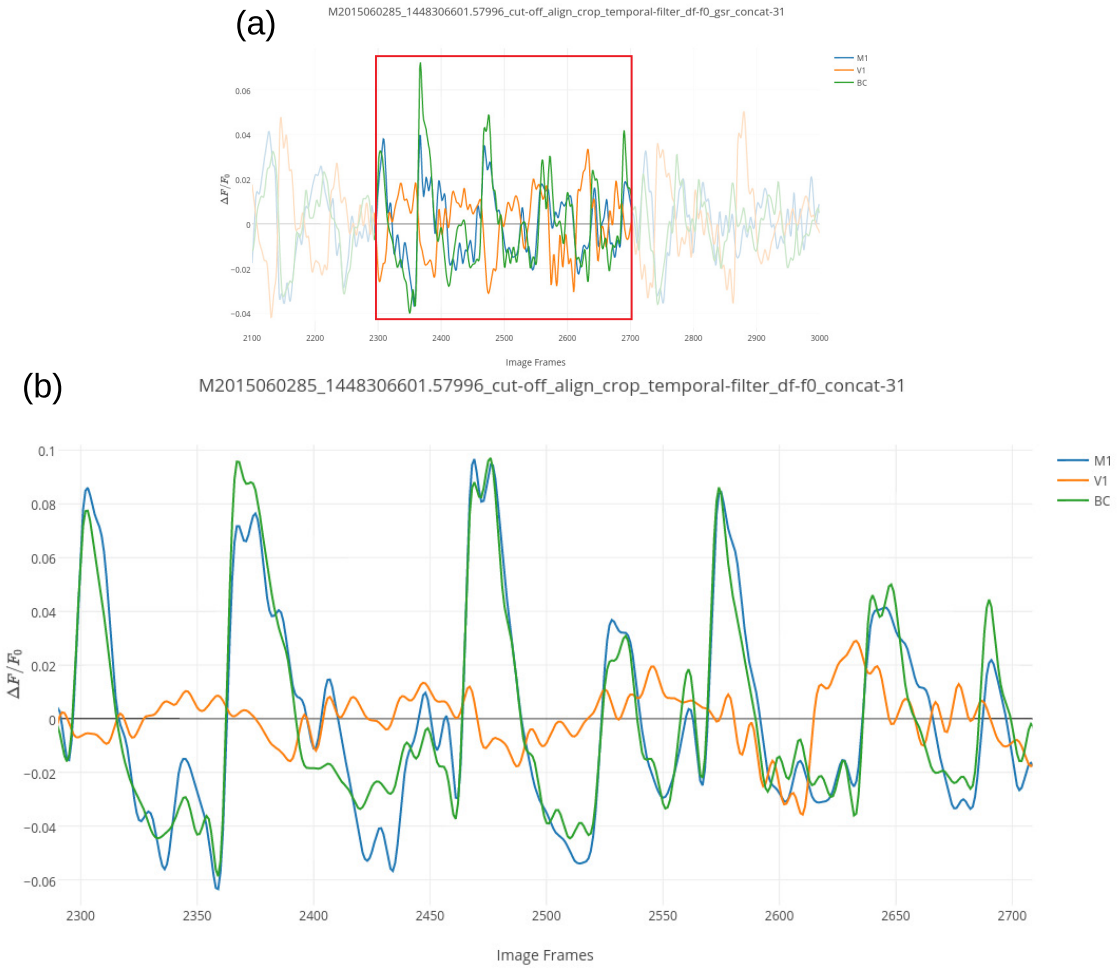


Figure 2.6: Time plots for selected ROI spontaneous activity without GSR. (a) Zoomed in segment of $\Delta F/F_0$ activity without GSR applied between. The 2100th and 3000th frames from all 31 30 second recordings concatenated of spontaneous activity from mouse #0285 (frame rate = 30Hz). $r_{M1-BC} = 0.905$ p ≈ 0, $r_{V1-BC} = 0.0739$ p $= 0.013\,359\,322$, $r_{M1-V1} = 0.144$ p $= 6.333\,26 \times 10^{-6}$. (b) Further zoomed in segment of $\Delta F/F_0$ activity between the 2300th and 2700th frames highlighting asynchronous activity between V1 (orange) against BC (green) and M1 (blue). Correlation coefficients for the full 30604 frame time course: $r_{M1-BC} = 0.895$, $r_{V1-BC} = 0.350$, $r_{M1-V1} = 0.345$

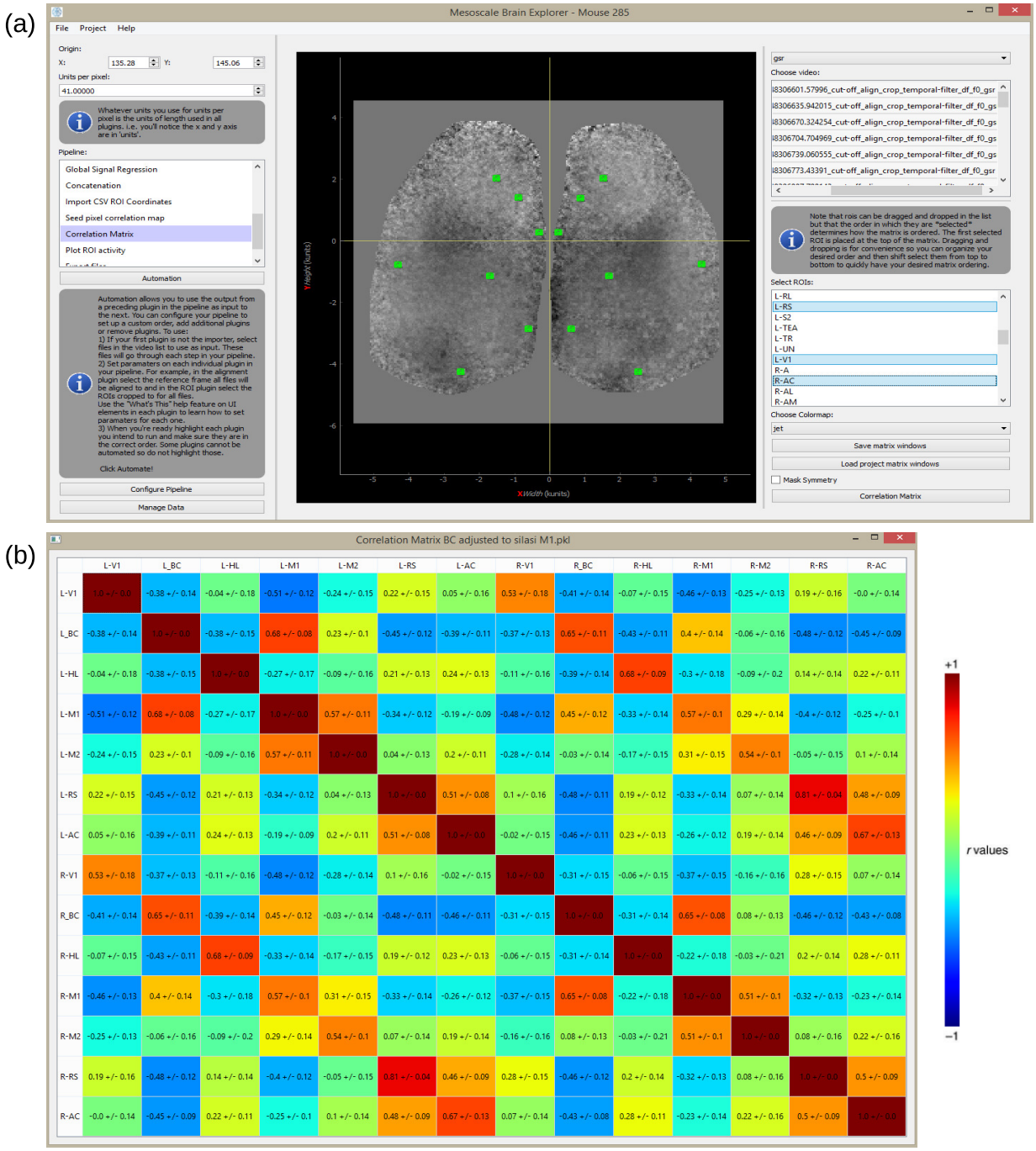


Figure 2.7: Correlation matrix for selected ROIs with GSR. (a) UI of the correlation matrix plugin from where ROIs are selected along with image stacks to generate connectivity matrices. A single image depicting instantaneous $\Delta F/F_0$ is shown for ROI placement (b) Mouse #0285 correlation matrix following collection of spontaneous activity via automated head-fix protocols. The data is presented in units of Pearson correlation (r-value) and the STDEV reflects variability of r-values between repeated 31 trials.

Plugin Name	Function
Average	Averages image stack activity into a single frame.
Alignment	Aligns image stacks against a user-defined reference frame.
Calculate df over f0	Computes fractional fluorescence change.
Channel Math	Computes arithmetic division, multiplication, addition or subtraction between image stacks.
Concatenation	concatenates images stacks into a single stack.
Correlation Matrix	Generates Correlation Matrix.
Create ROIs	Allows for the manual creation of polygon ROIs.
Crop Border	Crops a user-defined percentage of the total width of an image stack from all sides of an image stack.
Empty Plugin	Template plugin for developers.
Evoked Average	Averages selected image stacks across image stacks to create a single averaged image stack.
Export Files	Exports image stacks to .tif, .raw or .mp4
Global Signal Regression	Applies GSR.
Import CSV ROI Coordinates	Imports coordinates from .csv and creates square ROIs at those locations of user-defined width.
Import Image Stacks	Imports .raw, .tif or .npy image stacks according to user parameters.
Plot ROI Activity	Plots ROI activity across stacks for selected ROIs.
SPC Map	Computes SPC maps for selected image stacks and coordinates.
Set Coordinate System	Sets the origin for a single image stack; averages the origins over multiple image stacks to designate an averaged global origin for a project.
Shift Across Projects	Shifts files from another project to have the same origin as the project the user is working from.
Standard Deviation Map	Computes standard deviation maps which displays the variance in activity across frames for each pixel via a colourmap.
Temporal Filter	Applies a Chebyshev temporal filter with a user-defined bandpass.
Trimming	Discards frames from the start and/or end of selected image stacks.
Unsharp Filter	Applies an unsharp filter with user-defined kernel size.

Table 2.1: Table of available plugins with brief descriptions. See Appendix A for fuller descriptions of each plugin.

They can be inserted without any change to MBE or other plugins and without restarting the application, easing the development of plugins. This framework makes it easier for developers to exchange their own custom-made plugins without having to worry that another developer's setup may have compatibility issues.

It should be noted that the methods MBE provides are not without their limitations. Perhaps the most pressing limitation is that neither SPC maps nor correlation matrices provide information on connection directionality making causal inference unclear. This may be solved with the addition of plugins that perform Granger causality analysis (Anil K. Seth, Barrett, and Barnett 2015) thereby providing diagrams that include nodes for brain regions and arrows denoting the presumed directional flow of brain activity. Alternatively, experimentalists may opt for collecting non-spontaneous activity through techniques such as CHR2 stimulation, as has previously been undertaken by our lab (Lim, Majid H. Mohajerani, et al. 2012; Lim,

J. M. LeDue, Majid H. Mohajerani, et al. 2014; Lim, J. M. LeDue, and Timothy H. Murphy 2015). The application supports the use of evoke-triggered data through plugins such as the Evoked Average plugin (See appendix A.10).

2.6.1 Temporal Filtering

Temporal filtering spontaneous activity data has its own limitations. Applying a bandpass filter limits our sampling frequency. If the bandpass consists of a sampling frequency range that is too high, fast artifacts such as the mouse's heart rate are potentially picked up, reducing sensitivity to brain activity. If the bandpass is too slow artifacts such as hemodynamic processes are accentuated (Chan et al. 2015; Bumstead et al. 2016). Finally, GCaMP6 variants have different decay times following activity, in some cases limiting the range of frequencies that can be reported (T.-W. Chen et al. 2013). For studies with corresponding sensory-evoked data the exact range of a bandpass for spontaneous activity can be selected based on how well the filtered data compares with averaged sensory-evoked data. But for most spontaneous data associated sensory-evoked data is unavailable and therefore the frequency band is chosen a priori. For transgenic Ai94 GCaMP6 slow mice we recommend a bandpass filter of 0.3-3Hz with a frame rate of 30Hz as it shows specificity over green fluorescent protein (Timothy H. Murphy et al. 2016). For Ai93 GCaMP6 fast a 1-10Hz bandpass was used in a previous study with good specificity over GFP mice that lack functional signals (Silasi, Xiao, et al. 2016). Ultimately, this limitation is at least mitigated by MBE in that the interface allows the user to easily modify the filter range and users analyzing sensory-evoked data will not suffer from this limitation (See appendix A.10).

2.6.2 Comparison with Related Software Toolboxes

We here provide an overview of recently developed software toolboxes FluoroSNNAPP, Scintillate and Vobi One. Vobi One, like MBE, is a software package dedicated to the processing of functional optical imaging data (Takerkart et al. 2014). It is also written in Python and offers a roughly analogous architecture. The GUI likewise has a side-panel from where a user can follow progress or navigate to a particular "Process" which, just like a plugin in MBE, is a single script of code running that individual process. The application is likewise extensible, allowing users to add their own custom scripts and add them as "processes" to a custom pipeline. The application makes use of a "condition file" that summarizes info of all imported trials used by the processes and allows for interfacing with external software that cannot directly access BrainVISA (see following paragraph). This is analogous to MBE's JSON file which fulfills the same purpose. Whereas MBE provides importing routines for two commonly used versatile file formats .tiff and .raw, Vobi One provides importing routines for two file formats used by two popular CCD camera vendors - .blk files for Optical Imaging Ltd. and .rsd files for SciMedia USA Ltd. Both applications offer spatial binning, however Vobi One additionally offers temporal binning. As with MBE, upon import files are converted to a single file format that is used across all processes/plugins. Vobi One makes use of NifTI-1, a file format specifically made to foster interoperability at the file-exchange level between FMRI data analysis software packages. MBE, in contrast simply uses the standard binary file format (NPY) offered by the Python NumPy package (Walt, Colbert, and Varoquaux 2011). Nothing prevents either application from supporting file

formats of the other with both offering documentation for supporting additional importing routines.

The main point of departure between the two applications is that Vobi One is integrated with BrainVISA, whereas MBE is not. BrainVISA is an open source software platform that provides a complete modular infrastructure for different neuroimaging software. It organizes heterogeneous software and data and provides a common general graphical interface across pipelines for different applications. This can essentially provide a view where each software toolbox comprised of plugins is itself a plugin in BrainVISA. With this integration Vobi One offers cross-app automation. BrainVISA offers an iterate function allowing the same analysis with steps across toolboxes to be performed on different datasets - i.e. this sets up a loop from the GUI without having to write a program. This automation is much more comprehensive than MBE owing to its integration with BrainVISA. However, MBE does allow the user to string plugins in any order to produce a custom automated pipeline where all input files are processed through all steps in the pipeline. Instructions for this procedure are provided in the left side panel (Fig. 2.1a). Vobi One also offers three linear models for denoising optical recordings. The selected model is used to break down a recording into its noise and signal components and thereby extract the fluorescence response (Takerkart et al. 2014; Flotho et al. 2016). While Vobi One benefits from BrainVISA integration, MBE is much easier to set up because of its standalone architecture.

Vobi One is, to our knowledge, the only software toolbox with significant architectural and functional similarity to MBE. Two further recently published toolboxes, FluoroSNNAPP and Scintillate (Patel et al. 2015; Dublon et al. 2016), are related but are aimed at different end-users. FluoroSNNAPP is a MATLAB package for the automated quantification of single-cell dynamics and network activity (Patel et al. 2015). Nothing prevents MBE from being used to generate correlation matrices for cellular recordings and thereby quantifying single-cell dynamics and network activity. Both toolboxes offer $\Delta F/F_0$ and ROI drawing functionality. However, FluoroSNNAPP further integrates an automated cell identification method based on spatiotemporal independent component analysis and offers 3 methodologies for event detection: percentile based thresholding, wavelet transform decomposing a time-varying signal into frequency and time components, and template-matching using a database of known transient waveforms (Patel et al. 2015). FluoroSNNAPP is thus intended solely for comprehensive microscale analysis.

Scintillate is a MATLAB package that offers real-time $\Delta F/F_0$ while image acquisition is on-going, providing the user with signal change information and the means to further refine subsequent acquisitions (Dublon et al. 2016). Once signal change has been pinpointed, the user may change objectives, centre the image over that specific area or alter camera settings (Dublon et al. 2016). Scintillate is thus intended for use during data collection, whilst MBE is designed for data analysis after collection and is very appropriate for on the go analysis during an experiment by inexperienced users.

2.6.3 Conclusion

MBE provides a flexible software which is geared first to visualizing connectivity relationships within spontaneous activity data collected using widefield imaging. As a method-agnostic application MBE is well suited to being used to analyze data from brain activity indicators other than GCaMP, such as VSD (Majid H. Mohajerani, Chan, et al. 2013; Chan et al. 2015), glutamate-sensing fluorescent reporter (Xie et al. 2016)

or voltage-sensitive fluorescent protein (Carandini et al. 2015). The software is also applicable to intrinsic signal imaging (Y. Ma et al. 2016) formats and laser speckle imaging, or the flexible architecture can be extended to support any large image data set. While we have focused on mesoscale functional relationships within a single mouse, the approach could also be used for cellular GCaMP imaging (Winship and Timothy H. Murphy 2008) and the correlation based tools used to draw functional mapping between individual neurons and their neighbours. MBE also offers a "shift across projects" (see appendix A.18) plugin to align image stacks from different mice onto the same coordinate system, allowing for the generation of connectivity matrices averaged across trials from different mice. A simple division plugin is also included that applies division to selected image stacks. Importantly, dividing fluorescence F/F_0 by intrinsic reflectance F/F_0 can be used for hemodynamic correction (see appendix A.4) (Ron D. Frostig and Chen-Bee 2009; Y. Ma et al. 2016; Wekselblatt et al. 2016).

In conclusion, despite aforementioned limitations in the processing pipeline as well as with interpreting the end result, correlation matrices, SPC maps and standard deviation maps (see appendix A.19) provide simple and effective methods for identifying patterns of regional mesoscale functional connectivity changes. As an application that standardizes these approaches, that saves each processing step to file and keeps data organized, MBE should be a useful exploratory tool for any person performing functional connectivity analysis.

Chapter 3

Architecture and Rationale

3.1 Introduction

This chapter can be thought of as the "Methods" chapter and is primarily written to supplement the tutorial provided in Appendix C to make it easier for readers to modify MBE for their own purposes.

We will first cover an overview of the development framework chosen for MBE and then the different parts of MBE, providing labels for the different panels so as to have a point of reference. Next we will provide an overview of how the code is functionally organized with further reference to UI components that appear frequently in the application in particular panels. After establishing how code interacts to provide particular UI components and panels at a high-level we take a step-by-step approach and guide the reader through a tutorial where they add their own code to extend the applications functionality, thereby integrating new code with the current architecture.

3.1.1 A Simplified Introduction to Software Abstraction for Neuroscientists

Software abstraction allows for the ability to engage with a concept while safely ignoring some of its details — handling different details at different levels. Any time an aggregate is implemented, this is an abstraction. If your code models a “house” rather than a combination of glass, wood, and nails, this code is abstracting away the underlying details of the house. If further code models a collection of houses as a “town,” then this is another another abstraction (McConnell 2004). Encapsulation, inheritance and polymorphism are all essential principles of software abstraction.

Encapsulation

Encapsulation refers to the bundling of data with the methods that operate on that data (Rogers 2001). It is essential for modularizing subsystems and aids in long-term code maintenance as errors become easier to trace. Information hiding is an important form of encapsulation where the software developer hides design and implementation decisions in one place away from the rest of the program (McConnell 2004). This is typically done to encapsulate critical subsystems that should not be changed or to hide difficult design and implementation areas that might have been done poorly that might need to be done again. Information

hiding compartmentalizes these subsystems to minimize the impact their bad design or implementation might have on the rest of the application (McConnell 2004). Information hiding is one of the few theoretical techniques that has indisputably proven its value in practice, which has been true for a long time (Boehm 1987). Large programs that use information hiding were found years ago to be easier to modify — by a factor of 4 — than programs that don't (Korson and Vaishnavi 1986). Information hiding eliminates rework and is particularly effective in incremental, high-change environments (Boehm 1987) - such as those that define the Evolutionary Development Model (EVO) framework used by MBE, which is introduced in the next section. Encapsulation and in particular information hiding is therefore a key principle in MBE's design.

Inheritance and Polymorphism

Don't Repeat Yourself (DRY) is a principle of software development aimed at reducing repetition. This is especially valuable in systems developed under EVO that experience a high degree of change as it reduces "information bloat" which in the long-term makes a system harder and harder to maintain and modify. Inheritance and polymorphism are software abstraction principles that adhere to DRY.

Inheritance involves a submodule inheriting attributes or routines from a more general module. Inheritance simplifies programming because if we have a processing pipeline where each step is its own module but they all share common features (e.g. share the same graphical user interface) then these common features can instead be encapsulated into a more general module that each processing module inherits. Another advantage of this is that only one change in the general module results in a change to all processing modules that inherit the general module as they will all inherit the change. It should be noted that because inheritance exposes details of a general system to a subsystem inheriting it that this might break encapsulation and therefore violate information hiding (Gamma et al. 1994). There is therefore a trade-off to be made between inheritance and encapsulation.

Polymorphism is the provision of a single operation that can apply to various types. If a single coded operation can handle both integer multiplication and matrix multiplication then this operation is said to be polymorphic. Like inheritance, polymorphism simplifies programming by adhering to DRY. If a programmer wants to perform an operation on imported data that has a new structure from any other data analyzed thus far there may be a problem even if the operation is commonly performed on data with more commonly understood structure. A typical code-and-fix response would be to code a custom script from scratch. If the programmer however has routine that is suitably polymorphic it may be able to handle the new data immediately. Ultimately, encapsulation, inheritance and polymorphism standardizes how a program anticipates change. A study of great designers found that one attribute they had in common was their ability to anticipate change (Glass and DeMarco 2006). Accommodating changes is one of the most challenging aspects of good program design and an especially salient issue in research where analysis required may change unpredictably following unexpected results.

Summary

The standardization abstraction principles provide are not merely to offer robustly designed software but have implications for how the neuroscience community combats the ongoing replication crisis. The remark-

able growth in neuroscience over the past 50 years has led to smaller and subtler phenomena being studied, whereas before easily discernible phenomena were targeted for study first (Button et al. 2013). Dramatic improvements in technology has advanced the flexibility of research design and analysis (R. A. Smith et al. 2002) such that these phenomena can be observed, but the average sample size used to verify a finding with this new technology has not changed substantially over time (Vesterinen et al. 2011), despite the fact that neuroscientists are likely pursuing smaller effects. This increases the likelihood that statistically significant findings are spurious and may be the root of the replication failures in the preclinical literature (Prinz, Schlange, and Asadullah 2011). One proposed solution is improving the availability of materials and data. Leading journals are increasingly adopting policies for making data, protocols and analytic codes available (Eglen et al. 2017; Kitzes, Turek, and Deniz 2017; Sandve et al. 2013). However, these policies are uncommonly adhered to (Alsheikh-Ali et al. 2011; Eglen et al. 2017; Kitzes, Turek, and Deniz 2017; Sandve et al. 2013), and thus the ability for independent experts to repeat published analysis remains low (Ioannidis et al. 2009).

In standardizing how MBE adapts to changes in code by the three outlined principles of abstraction MBE makes sharing analysis easier. Programmers wishing to repeat a particular analysis with slight modifications can more easily do so as the code for MBE can be more easily extended. This complements role of MBE as offering ease of proofreading for high-throughput pipelines. Moreover, by offering a GUI MBE also makes it easier for researchers to repeat another researcher's analysis on their own data.

3.1.2 Development Framework

Software development is best approached as a continuous incremental integrative process (McConnell 2004). As such, the framework selected for development can hugely impact the long-term efficacy of an application. Most research-grade code written by labs tend to make use of the code-and-fix model (Boehm 1988). This is the basic model used in the earliest days of software development contained in two steps: 1) Write some code. 2) Fix problems in the code. Thus, the order of the steps was to do some coding first and to think about the requirements, design, test, and maintenance later. For small-scale scripting problems this model is not only adequate but ideal due to the pragmatic focus on results. For instance, many labs find themselves wanting to output a single plot a particular way based on some sample data. Writing a single script in MATLAB to do this makes use of this model and is, in most cases, the best approach.

This model however has three primary difficulties that was noted by early programmers which projects are scaled up (Boehm 1988). 1) After a number of fixes, the code became so poorly structured that subsequent fixes were very expensive. This underscored the need for a design phase prior to coding. 2) Frequently, even well-designed software was such a poor match to users' needs that it was either rejected outright or expensively redeveloped. This made the need for a requirements phase where the user requirements to be met are rigidly laid out prior to the design phase. 3) Code was expensive to fix because of poor preparation for testing and modification. This made it clear that planning in the early phases how to write code in a way to make future testing and evaluation easy was needed.

The explicit recognition of a need for more constrained orderly development led to the stagewise and waterfall models. The waterfall model was highly influential in the 1970s, providing refinements to the

stagewise model that at early as 1956 stipulated that software be developed in successive stages (Boehm 1988). The waterfall model further provided feedback loops between stages, but as might be expected from the name, the framework deliberately made it harder for feedback to backtrack upstream.

For some classes of software, such as secure operating systems, the most effective way to proceed is through elaborated documents as completion criteria before progressing to the implementation phase. However, even with extensive revisions and refinements, the waterfall model's basic scheme is fundamentally ill-suited for projects that require a flexible framework such as interactive end-user applications like MBE that may require unpredictable changes following user feedback. Document-driven frameworks have pushed many projects to write elaborate specifications of poorly understood user interfaces, followed by the design and development of large quantities of unusable code (Boehm 1988).

The EVO evolutionary development model (Gilb 1988; May and Zimmer 1996) was developed to combat both the lack of structure of the code-and-fix model as well as the lack of flexibility of the waterfall model. The EVO model is well suited to situations in which users say "I can't tell you exactly what I want, but I'll know something is what I want when I see it" (Boehm 1988) which is a situation MBE and interactive end-user applications often find themselves in. Target users do not accurately introspect about their own data analysis and visualization needs, making these needs invisible to them. This is a well-known phenomenon in psychology (Anders and Simon 1980). This is the primary reason this framework was adopted to implement MBE. The EVO development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product and provide feedback at the end of each cycle (Fig. 3.1) (May and Zimmer 1996). The inevitable change in expectations when users begin using the software system is addressed by EVO's early and ongoing involvement of the user in the development process. EVO thus attempts to strike a balance between waterfall and code-and-fix by offering a structured, disciplined avenue for flexible experimentation (Boehm 1988). EVO is also similar and arguably a form of Agile development, which is a development framework largely popular today. We selected EVO and not Agile due to Agile's focus on roles for each team-member, which is untenable as MBE was primarily developed by a single individual.

Nonetheless, EVO also has its difficulties. It is based on the assumption that a developer's environment will be flexible enough to accommodate unplanned evolution paths. This assumption makes it look suspiciously like the code-and-fix model as EVO emphasizes tackling hard-to-change code first before addressing long-range architectural considerations such as modularizing a systems appropriately for later subsystem integration (Boehm 1988). For instance, when several independently evolved subsystems need to be integrated into a single system preplanning for such an integration long before developing each subsystem is crucial. EVO however pushes developers to develop prototypes of each subsystem to obtain user feedback by the end of a cycle (Fig. 3.1) (May and Zimmer 1996).

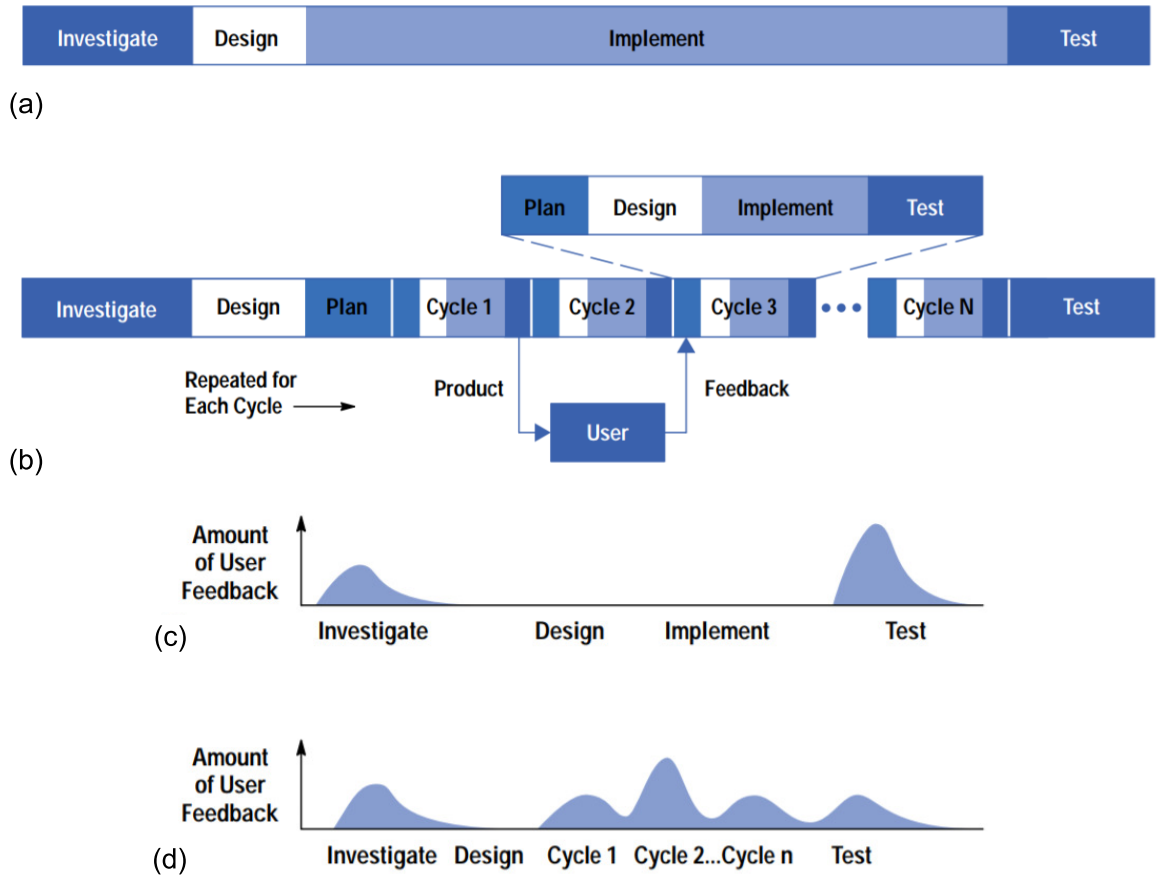


Figure 3.1: Software development life cycles. (a) Traditional waterfall model. (b) Evolutionary development model. Amount of user feedback during (c) the traditional waterfall development process and (d) the evolutionary development process (EVO). In: Elaine L. May and Barbara A. Zimmer (1996). "The Evolutionary Development model for software". In: *Hewlett-Packard Journal* 47.4, p. 39. ISSN: 00181153. URL: http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=9608302686&site=ehost-live&scope=site (visited on 05/26/2017). © Copyright 1995 Hewlett-Packard Company. Figures coped by permission of the Hewlett-Packard Company.

.

3.2 General Overview

3.2.1 Main Window and Interface Orientation

Fig. 3.2 shows a schematic diagram of MBE's interface. The interface is divided into two parts, the *Pipeline Pane* on the left (red) and the *Plugin Panes* on the right (blue).

Each processing step in MBE is independently contained within a plugin which itself is contained within a single script. The *Plugin List* in the centre of the Pipeline Pane contains a plugin list where plugins can be selected, bringing the selected plugin into view in the Plugin Panes. Each session of MBE requires its own "project." The *Project-wide Settings* in the *Pipeline Pane* contains UI components that can be used to modify

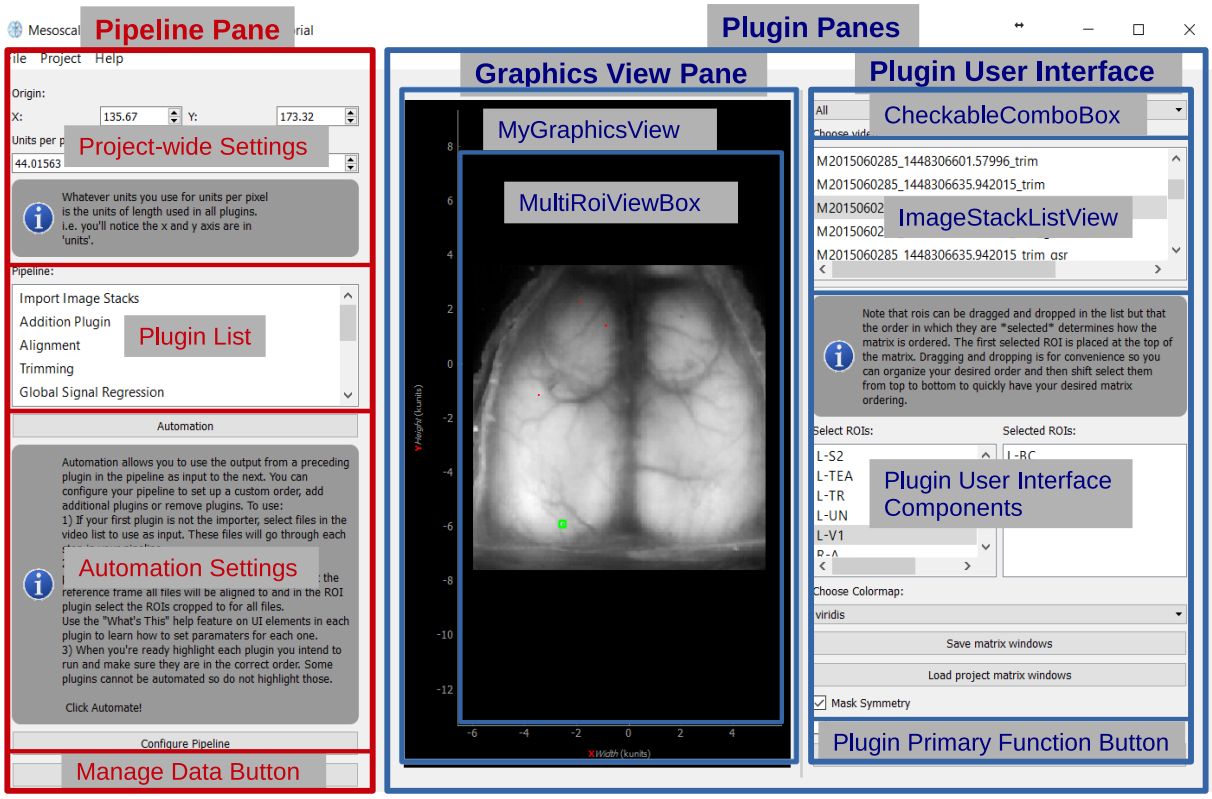


Figure 3.2: Schematic diagram of panes and UI component locations in MBE.

parameters that are used in a session across plugins. For example, the project's coordinate axis origin and the width per pixel needs to be specified by the user here as these parameters are used across all plugins. Further UI components added that likewise all plugins make use of will be added here. The *Automation Settings* in the Pipeline Pane contains instructions and UI components necessary for automation. Automation allows the user the option of creating a processing pipeline. Plugins are included and ordered via the Pipeline Configuration window (Haupt et al. 2017) (See Chap. 2, Fig. 2.1). This makes them available for selection in the plugin list. When automation is activated data is funnelled through each selected plugin and processed in each based on the parameters set in the plugin panes by the user for each plugin. In some cases what parameters must be set for automation may not be clear. Consequently, all plugins contain contextual help messages to guide the user. These can be accessed via clicking help and "What's this?" and then clicking on the UI component of interest. The *Manage Data Button* opens a *FileTable* (See Fig. 3.4) where all project data and metadata can be viewed.

The Plugin Panes consist of two panes (in most cases) - the *Graphics View Pane* in the centre that displays an interactive graphic of the image stack under analysis and the Plugin UI pane on the right that contains all the UI components of a particular plugin. The Graphics View Pane consists of classes that together are used to display interactive graphics (See next section). The Plugin UI on the far right typically follows the same structure. First there is a drop down list with checkable options for filtering image stack selection called the *CheckableComboBox*. This is followed by a list of image stacks that the CheckableComboBox can be used

to filter. From this list the user selects image stacks to operate on. This list is called the *ImageStackListView*. Following this we have UI components (e.g. dropdown lists, checkboxes, sliders etc) specific to the plugin that are used to operate on the image stacks selected in the ImageStackListView. Finally, at the bottom of the Plugin UI pane there is a button which executes the plugin's primary function. Note that although this is the standard outline for most plugins, that there are and can be exceptions.

3.2.2 Visualization Window

Some plugins output interactive visualizations instead of processed image stacks. These interactive visualizations are organized in a general-purpose "visualization window." Such a window consists of Dock widgets. Each dock can be dragged by its title bar to occupy a different space within the window. Docks that are dragged on top of one another are stacked in a tabbed layout. Additionally, the borders between docks may be dragged to resize, a dock can be double-clicked to place it in its own window and docks can be individually closed without closing the window. The configuration for a dock along with the data comprising the graphics can be saved to file or another visualization window loaded.

This docked paradigm is designed to offer freedom in how visual data is displayed and organized, allowing for more effective exploratory analysis to a broad audience.

As an example of use, consider Fig. 3.3. The top window contains 9 docks of SPC maps (numbered for convenience) where each dock has multiple image stacks each with the same seed. Docks 1 and 2 show SPC maps with seeds at R-V1 (1) and L-V1 (2) with the crosshair placed by the user in the hemisphere opposite to the location of the seed at V1. Inter-hemispheric functional connectivity can thus be analyzed. Docks 3-4 display SPC maps with seeds at BC, also with the crosshair in the hemisphere opposite to the seed. However in this case the SPC maps are of two different image stacks and the crosshair is in the same location for each. Correlation at similar regions for different image stacks can thus be analyzed. Note further that dock 4 contains one tab and was dragged there from dock 3. Dock 3 has also been zoomed in on, an interactive feature of each graphic. Docks 5-6 simply illustrate that the crosshair can be removed and are included for completeness. Dock 7 shows the menu of actions available to a user including exporting an image after manipulating it as they see fit and resetting the image to its original size. Dock 8 with the seed at L-HL shows a different coordinate (R-V1) loaded and overlaid with the image. Coordinates can be loaded in any image to aid in orientation. Finally in the final dock 9 the user has clicked the help button to bring up instructions on how to interact with the visualization window and its docks.

Docks A-D likewise show various interactivity available to the user after outputting the visualization window from the activity plots plugin. Docks A-B are activity plots of V1 used in docks 1-2. Here we can for instance compare activity in on hemisphere vs the other as well as across image stacks. In docks A and B there is a fourth green line absent in the others. This is an average line that is easily added via transforms available in the right-click menu. As such the average of activity in all 3 image stacks in the L-V1 can easily be compared with the average of the same 3 image stacks in R-V1. In dock C a fast Fourier transform transform has been applied to the HL activity to show the power spectrum. Finally in dock D the user has zoomed in on particular peaks in L-BC. This may be done for instance to compare activity between two image stack in L-BC to corroborate findings with docks 3-4 which are SPC maps of L-BC. Each plot can be

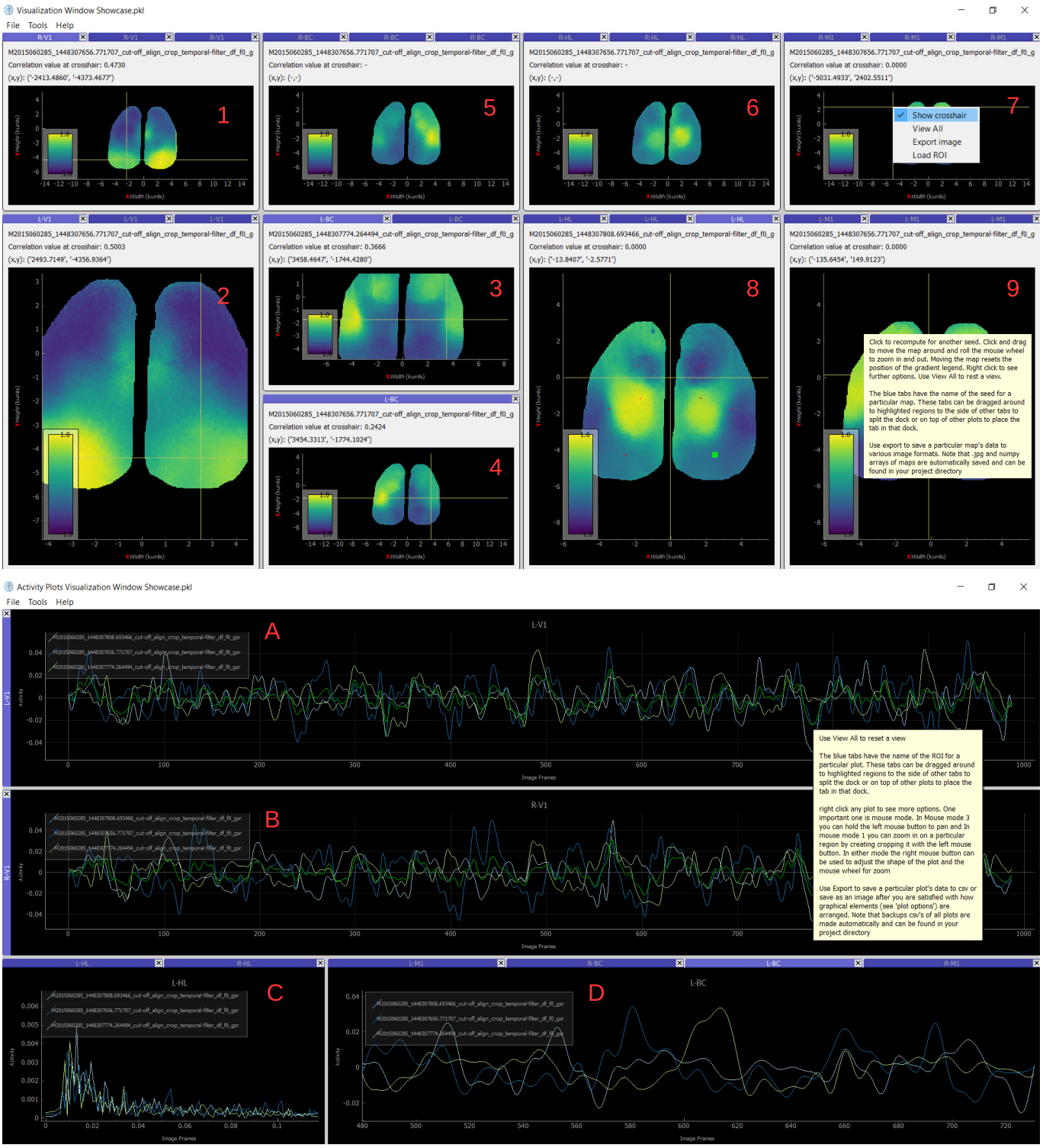


Figure 3.3: Visualizations outputted by the seed pixel correlation map plugin (top window, docks 1-8) and the ROI activity plot plugin (bottom window, docks A-D)

zoomed in on and activity easily exported. Just like the SPC maps visualization window the activity plots visualization window also has instructions in how to operate the docks and interact with the visualizations if the user presses the "what's this?" option under the help menu.

3.3 Code Organization

As of version 0.8.0 (See Appendix B) the source files that make up MBE are organized as in Fig. 3.5. The *mkl_dlls* folder contains .dll files necessary to freeze the code into a standalone executable. The *pics* folder contains all the images used by MBE. The *plugins* folder contains all plugin scripts where each plugin is contained within its own script. This folder also contains an *examples* folder where examples used in tutorials are kept (see Appendix C) and the *util* folder. The util folder (short for 'utility') contains high-level classes and functions that are used or could be used by multiple plugins. These are organized into the following scripts:

- **constants.py:** acts as a placeholder for variables that always have constant values.
- **custom_pyqtgraph_items.py:** the natural unit of code decomposition is the module, not the class as in languages such as C++. As such, all custom made UI items that inherit from the pyqtgraph module (e.g. QMenuCustom, GradientLegend in Fig. 3.4) are contained in this script.
- **custom_qt_items.py:** all custom made UI items that inherit from the pyqt4 module (e.g. FileTable, ImageStackListView, InfoWidget, RoiList, CheckableComboBox, MyTableWidget, PlayerDialog in Fig. 3.4) are contained in this script.
- **debug.py:** a utility script for setting traces for debugging.
- **file_io.py:** contains all functions used throughout plugins for loading and saving to file. This includes checking system available memory and modifying duplicate path names such that a user's old files are not overwritten upon rerunning a process on the same file.
- **fileconverter.py:** contains functions for converting files from raw and tif formats to numpy arrays. Further file support would be implemented here and called by the fileimporter plugin
- **mygraphicsview.py:** Contains the *MyGraphicsView* class, a re-implementation of the GraphicsView class from the pyqtgraph module (Campagnola 2016) which serves as a container for a graphical scene used in most plugins where 2D graphical items can be rendered and interacted with (zoomed, panned, exported). MyGraphicsView extends this functionality by the addition of coordinate axis units based on user-input (pixel width determined by camera specifications) and the crosshair that can be used to locate anatomical regions of interest based on their position in 2D space. MyGraphicsView has a MultiRoiViewBox (See Fig. 3.4) which handles the graphical scene with added ROI creation properties .
- **parmap.py:** contains code for parallelizing the Python standard map function. This was modified from the parmap module (Oller 2016) and was planned to be used to parallelize code where possible

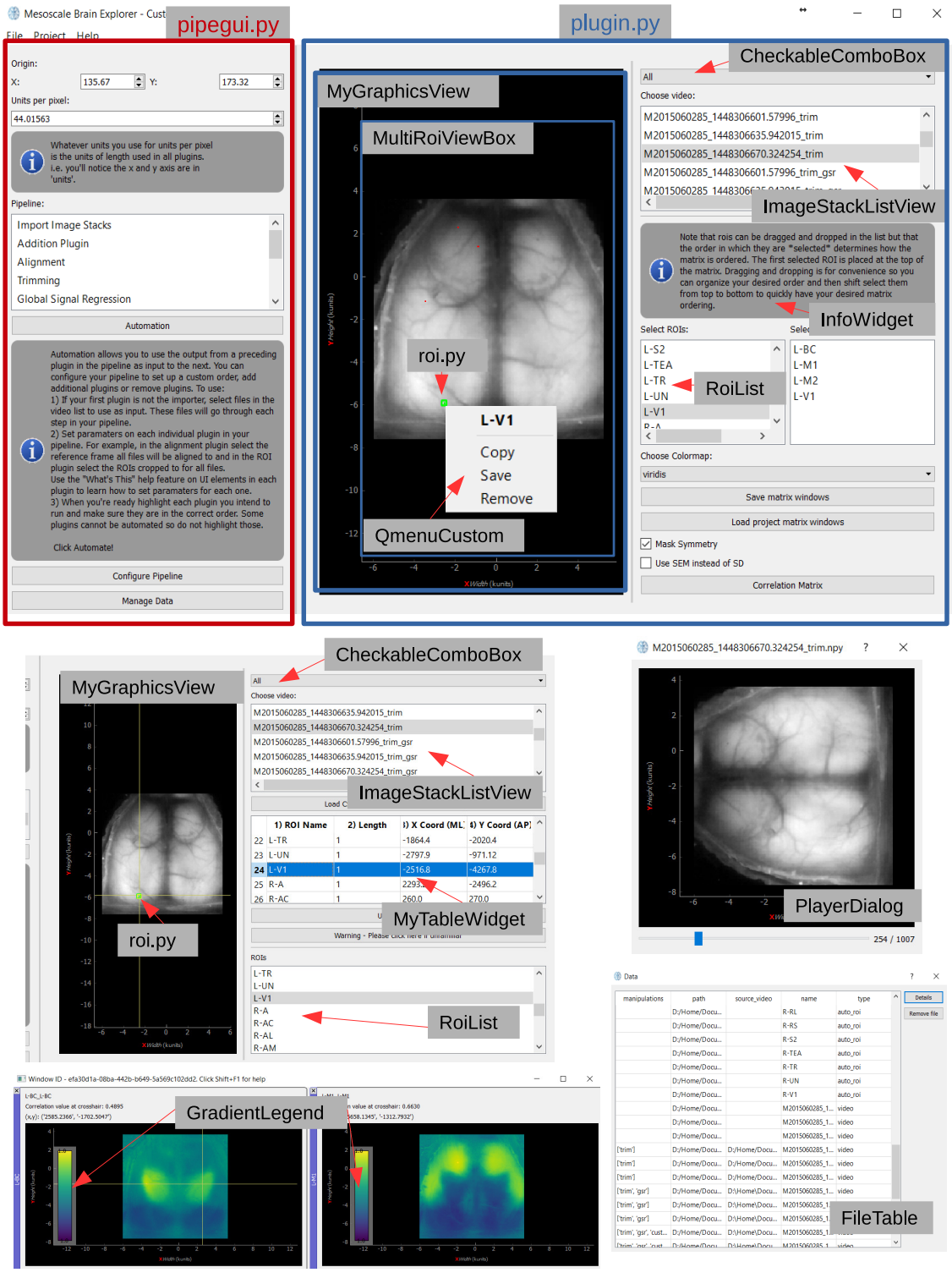


Figure 3.4: MBE screenshots annotated.

(specifically temporal filtering) to improve runtime. This optimization has not yet been realized and as of 0.8.0 (See Appendix B) remains an area of speculative development.

- **plugin.py:** This script contains the default behaviour of all plugins that inherit its classes, namely *DefaultWidget* and *DefaultPlugin.* The DefaultWidget class creates a CheckableComboBox and an ImageStackListView and places it at the top of the plugin UI pane. A primary execution button is also provided and added to the bottom of the plugin UI pane. This class also creates the MyGraphicsView for the scene and places it to the left in the graphics view pane (See Fig. 3.2). The DefaultPlugin class has functions that define a plugin's default automation behaviour. Plugins cannot be automated by default thus this class must be overridden by plugins that have automation behaviour (see Appendix C). Plugins that inherit these classes from plugin.py therefore automatically attain a standard functional layout and only have to have the plugin user interface components implemented (See Fig. 3.2).

- **project_functions.py:** contains functions commonly used across plugins. Most functions revolve around saving and updating metadata in the project json file *mbeproject.json*.

- **roi.py:** contains the ROI class which extends the ROI class in the pyqtgraph module with custom handles (defined by the Handle class) that allow ROIs to be drawn over each other which is typically required when cropping the two brain hemispheres. Altogether, roi.py is used to define the creation of user specified polygon ROIs of arbitrary size and shape for functions such activity plotting at the ROI or cropping to the selected polygon ROIs across a stack of images. These ROIs can be saved to file to a custom ".roi" format to be imported into a different project (see sect. where "project concept" is introduced). An example ROI is seen in Fig. 3.4.

- **viewboxcustom.py:** contains *MultiRoiViewBox* which extends the pyqtgraph module's ViewBox class so that it can additionally support the user-defined addition of multiple ROIs as defined by roi.py. Functionality inherited from ViewBox include internal scaling/panning of the image by mouse drag, scaling of contents by mouse or auto-scale when contents change and item coordinate mapping methods - used to map location of ROIs (Campagnola 2016). Code for viewboxcustom.py and roi.py was adapted from Micheal Hogg's open-source BMDanalyse, a program designed for the regional analysis of bone mass density through interactive visualization using the same frameworks as MBE uses (Hogg 2013).

- **visualization_window.py:** contains *DockWindow* an empty window with docks and save and load functionality. Much in the same way as the default classes in plugin.py, DockWindow is generally inherited by any class implemented to be the visualization window for a particular plugin. *DockWindowSPC* in spc_map.py and *DockWindowPlot* in roi_activity_plot.py inherit DockWindow to output visualization windows for their respective plugins. The visualization windows outputted by these two plugins have very different functions (Fig. 3.3 and Sect. 3.2.2) and as such each overrides save and load functions from DockWindow to define its own import and export protocols. Additional functionality that is applicable to all visualization windows across all plugins (e.g. more tools under the "tools" menu) is implemented within this script.

The *templates* folder contains JSON files used to initialize the settings of MBE during a session. Currently only one template JSON file, mbeproject.json, exists, which is used to set up a basic pipeline when a new project is created. In MBE each session must be run on a Project which is created or loaded upon startup via the Project class in project.py. Each Project object is defined by its own folder and JSON file defining its plugins (which make up its Pipeline List (See Fig. 3.2)), files that have been imported (image stacks, ROIs etc) and plugin parameters. Plugin parameters are the values the plugin UI components take. These values are updated to mbeproject.json when the user alters the parameters for a plugin (Haupt et al. 2017) (See Chap. 2, Sect. 2.2.1). When the Project is loaded at a later time those plugins are loaded along with the plugin parameters which are used to set the values of each plugin's UI components to their previous values.

The *datadialog.py* script contains a number of classes that bring up different dialogs after pressing the manage data button in the pipeline pane (Fig. 3.2). The primary window makes use of a FileTable object from custom_qt_items.py (Fig. 3.4) to display all data that comprise a Project and metadata that is associated with each image stack, roi or other data object.

The *pipeconf.py* script defines a a window that opens to allow a user to set which plugins are available and in what order they appear in the pipeline list (Fig. 2.1). *qtutil.py* contains custom UI-related functions (as opposed to classes that define actual UI elements as in custom_qt_items.py) for code that is often repeated such as error/warning/info message pop-ups. Finally, *pipegui.py* is the entry point for MBE. This script is primarily responsible for setting up the Pipeline Pane, adding plugins to the plugin list based on info stored in the project JSON file, checking whether automation criteria has been met for each plugin before allowing automation to proceed and funnelling data through the pipeline to be processed at each step based on parameters set at each plugin by the user.

hook-plugins.py and *pipegui.spec* are both, like the mkl_dll folder, used to freeze the source code to create a standalone executable. The pipegui.spec is the specification file that tells the installer how to process the source code. It contains instructions that that tell the installer to copy the mkl_dll folder's contents to the appropriate location and to look at the hook-plugins.py script to find all the files required to add the plugins to the executable.

3.4 Code Verification

To verify that MBE returns results that are correct we reproduced a correlation matrix produced by MBE using a script written in MATLAB that is known to output the correct result. This MATLAB script is what researchers in our lab would use if MBE were not available. The correlation matrix was made using data from mouse #285 and an identical pipeline of processing steps as in Fig. 2.2 and as described in Sect. 2.4. Instead of STDEV, standard error of the mean was used instead simply because this is more often used that standard deviation by researchers in the lab. The only other difference from Sect. 2.4 is that GSR was omitted because MATLAB and Python appear to have different parameters set for the package that handles GSR in either case. Slightly different values are therefore expected if GSR is in both pipelines, which doesn't provide a strong case for verification. However, with GSR omitted we would expect exactly identical values in the correlation matrices.

This is exactly what was achieved. As seen in Fig. 3.6, exactly identical values are obtained for each

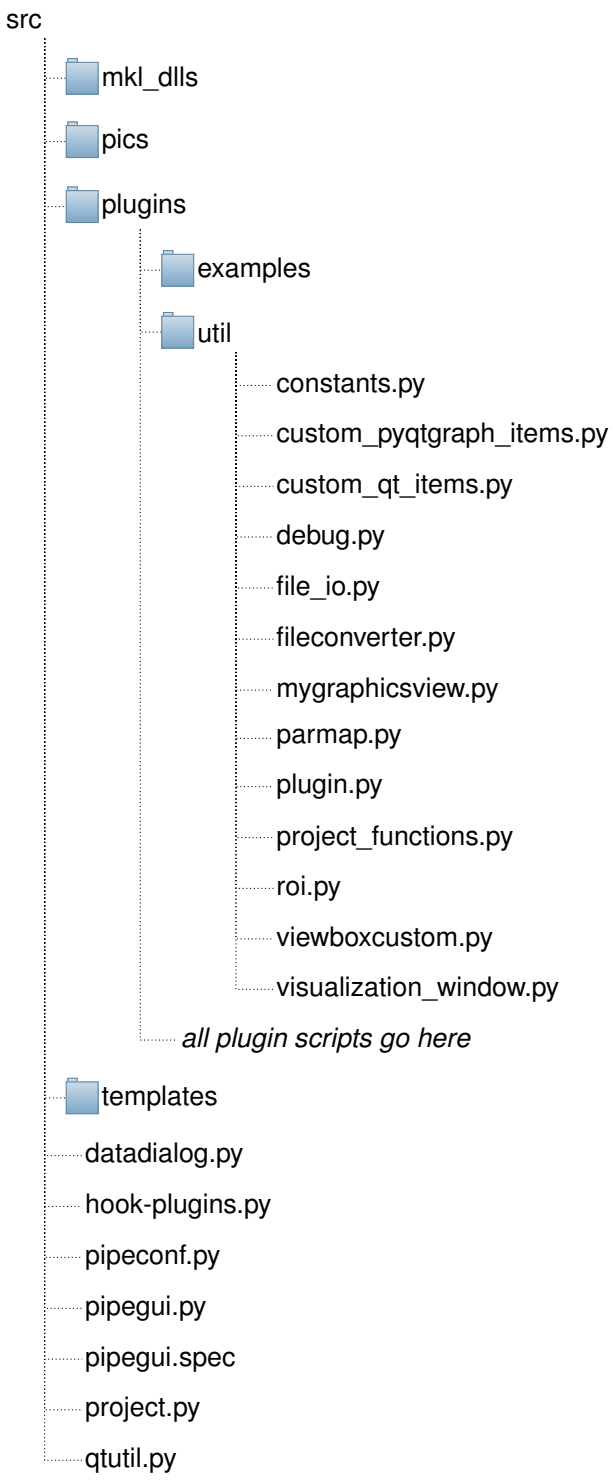


Figure 3.5: MBE file hierarchy.

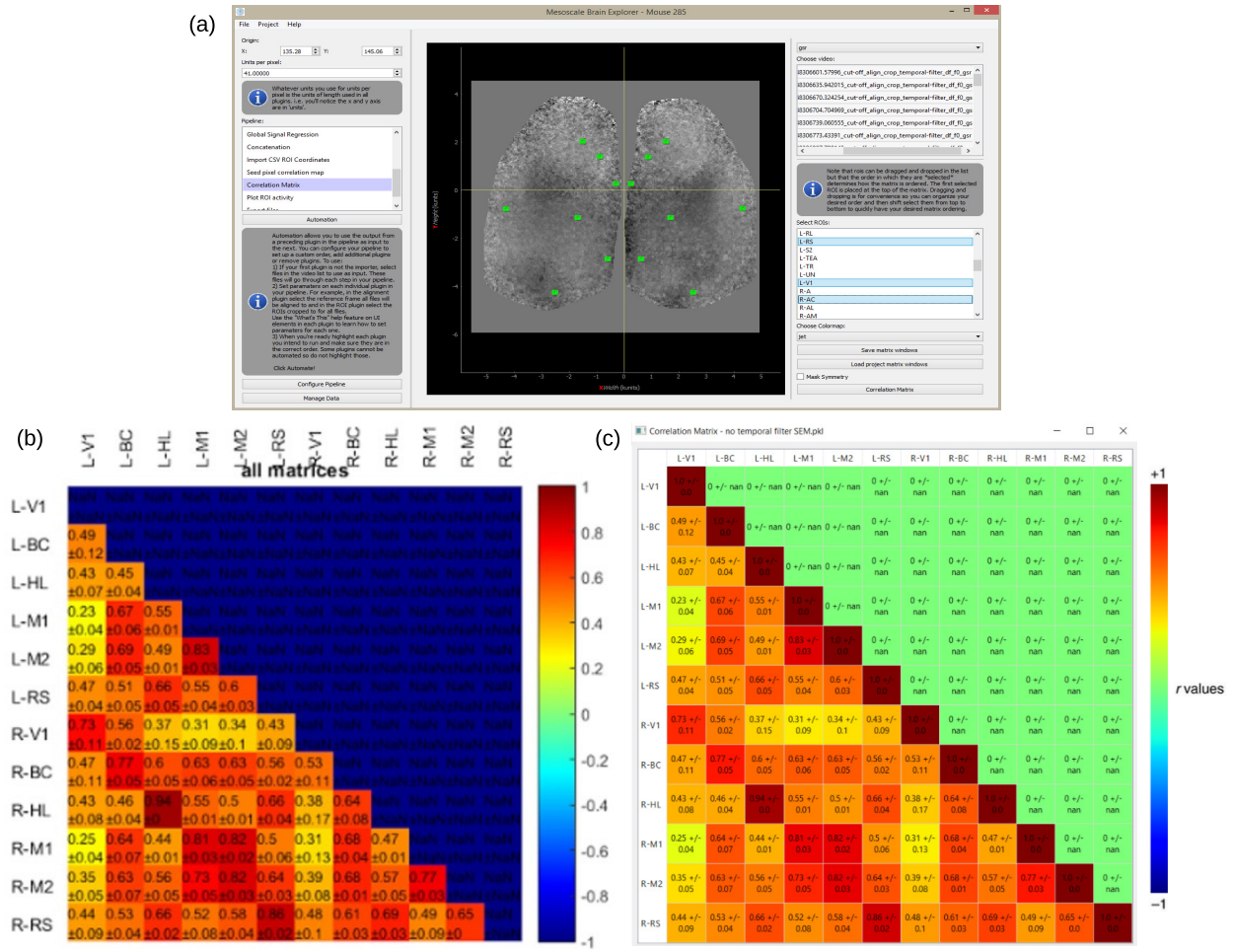


Figure 3.6: Code verification by comparing MATLAB vs MBE correlation matrix output.

ROI-ROI pair and each standard error value. This provides strong evidence that each step from data import, to coordinate setting to all the steps through the pipeline are working as required as any error along the pipeline would cause at least one of these ROI-ROI pairs to deviate. Brain activity plots and SPC maps use the exact same set of steps, except for the last step (See Fig. 2.2) that determines which visualization is produced. Therefore, this correlation matrix verification provides strong evidence that brain activity plots and SPC maps are also producing the correct output. Upon manual inspection of an arbitrary set of activity plots and SPC maps, they were judged by eye to be identical for MATLAB vs MBE.

Further testing would be required to fully verify MBE's code. This could be done through the implementation of code testing as discussed in Sect. 4.2.1.

3.5 Design Rationale

Designing MBE principally consisted of designing the user interface and deciding how UI elements will interact and designing the code organization such that it meets familiar software engineering standards. A

piece of software with a brilliant interface will be hampered by poorly designed code organization as it will be difficult to maintain. A piece of software with code that thoroughly abides by software engineering standards is hampered if its interface is terribly designed and hard to use for its intended users. We will differentiate between these two aspects of design as internal (code organization) and external (user interface) design.

3.5.1 External Design

The planning stage in the EVO framework discussed in Sect. 3.1.2 is related to what is called requirement analysis in software engineering (Kovitz 1998), which is directly linked to talking with and observing domain experts. The EVO framework MBE was implemented under is iterative and cyclic: the expert speaks and the researcher listens, the researcher abstracts, then elicits feedback from the expert on the abstraction (Sedlmair, Meyer, and Munzner 2012). Computational reproducibility and transferability guide how feedback influences MBE's external design as they are the goal of achieving an application where our first research aim is met. Computational reproducibility refers to a research project where a second investigator - including the original researcher in the future - can recreate the final reported results of the project, including key quantitative findings, tables, and figures, given only a set of files and written instructions (Kitzes, Turek, and Deniz 2017). Note that computational reproducibility is not the same as theoretical reproducibility of MBE itself. The measure of MBE's success is not that a different software engineer would design the same system, it is that the particular researchers it is designed for find it useful (Sedlmair, Meyer, and Munzner 2012). This is referred to as transferability and takes precedence as meeting the needs of the end-users requires intrinsically subjective field work (B. Brown, Reeves, and Sherwood 2011).

MBE's external design, in this context, focused on the feedback received from two domain experts who take the two critical collaborator roles in the design of research-grade software. One is the domain end user doing the actual analysis and is the person using the tool called the front-line analyst (Sedlmair, Meyer, and Munzner 2012). In MBE's case this person is a graduate student with limited expertise in programming. The second is the person with the power to approve or block the MBE project, including authorizing people to spend time on the project. This person is called the gatekeeper (Sedlmair, Meyer, and Munzner 2012) and is the principal investigator of a neurophotonics lab. Starting with a design before contact is established with at least one front-line analyst is a known pitfall in the design of software in academia (Sedlmair, Meyer, and Munzner 2012). As such, contact was established with the front-line analyst long before the final product was released. Feedback was elicited that greatly influenced the course of MBE's external design. Spatial filtering did not initially have a range where the user can choose to process a subset of an image stack instead of all frames. This was done due to only a few frames being required for alignment to work properly. Alignment, processed by another plugin, remains the only way that spatial filtering is used. MBE was initially set on a paradigm where each plugin would only serve one function to keep the design lean. A fully spatially filtered stack could first be trimmed and then used in alignment. However, alignment often included setting of spatial filter parameters to different values after aligning and realigning to test if alignment is better. This occurs frequently and the amount of time needed to first spatially filter and then trim proved impractical when a quick and simple solution of letting the user specify the range of frames to

be spatially filtered would bypass this problem entirely. Despite now being considered an essential feature, this range-setting UI component was not originally planned for implementation since, in theory, MBE is functional without it. In the standard deviation map a log scale feature was specifically implemented to aid the front-line analyst's specific needs and not because it was believed to be broadly useful. It should thus be noted that more development and testing has gone towards certain functions than others due to their demands. The alignment plugin has by far the most complex user-interface. It was initially designed to be simple with most alignment decisions handled by set parameters and the imreg_dft module (Tyc and Gohlke 2016). The increase in complexity in the UI arose due to the evolving needs of the front-line analyst. Later data required more rigorous alignment demanding more user control leading to a more complicated interface that as of 0.8.0 (See Appendix B) is planned to become far more complex. Other plugins, by contrast, are as barren as only having a single button. The GSR plugin is such an example. A final point to note here is that MBE has thus been designed to be computationally reproducible, but the emphasis has been on making it easy to do so for the front-end analyst with the assumption that the front-end analyst would be able to use MBE's plugin architecture to more easily convey to other researchers how they could use MBE to reproduce their processing pipeline and results. We however have no empirical data to support that MBE does indeed aid in making reproducibility easier. This would require a case study of multiple front-end analysts, gathering data where their ability to reproduce each other's results is assessed when using MBE versus standard techniques. No such data has been acquired. The plugin architecture the entire conception of MBE is built on is not validated as the "best" external design. The design was found to work for the front-end analyst and thus no further attempt was made to find a better design.

A central question to MBE's external design is whether there is a real need or whether existing approaches are good enough. If current approaches are sufficient then domain experts are unlikely to go to the effort of changing their workflow to adopt a new tool such as MBE, making validation of the benefits of MBE's design difficult to acquire. This is a known pitfall in the development of software of academic research (Sedlmair, Meyer, and Munzner 2012). Automation (See Appendix C) suffers the most in this regard as it has not seen widespread use. This has been attributed to the requirements of the front-end analyst's data where intermediate steps consistently require manual catering insofar as even different image stacks require different processing within the same plugin (e.g. trimming each image stack differently instead of applying a single trim in bulk to all image stacks). The front-end analyst does make use of the feature in segments of her processing pipelines, but this is more of a convenience feature rather than a defining feature that frees up most of her time during the processing stage. This helps explain why automation in MBE is built in unconventionally and perhaps unintuitively. You would expect that instead an automation button would open a separate window where various automation parameters could be set. A prototypical minimalistic automation functionality was built on top of the interface that was already in place - as per the EVO implement phase (Fig. 3.1). Subsequent feedback revealed this feature worked adequately for the user's needs, but was not found to be as large of an advantage as expected. As such, further development on it was not pursued.

MBE's design does not solely take into account the desires and needs of the user in its external design. The default colourmap of MBE for instance is not jet, which is typically preferred by users. The jet colourmap is limited in being perceptually non-uniform, converging, and not sequential (Borkin et al. 2011;

Moreland 2016). It is perceptually non-uniform, meaning it is not clear to the human visual system what magnitude of increase there is between certain areas in the scale. In the green region of the colourmap there is a large segment that, although physically increasing monotonically, the visual perception system perceives as mostly uniform. Significant differences between "green values" in correlation maps or correlation matrices are therefore potentially masked. Other perceptually uniform regions on the jet colourmap such as the yellow band are much narrower, potentially accentuating insignificant differences between values nearby the yellow band. Jet can therefore mask details and de-emphasize extremes. Jet is not diverging. A diverging colourmap has changes in lightness and possibly saturation of two different colours that meet in the middle at an unsaturated colour. This is a property recommended of colourmaps when information being plotted has a critical middle value, such as correlation maps where data deviates around zero (Moreland 2016). Jet is not perceptually sequential. The number of hues used and the transitions between hues are not perceived by the human perception system as incrementally increasing, making jet a poor candidate for measuring an ordered range such as a correlation coefficient. Jet is also not usable for colourblind scientists. Borkin et al. (2011) have found that simply changing the jet colourmap used by domain experts in heart disease diagnosis to a colourmap that meets all three criteria above increased correct diagnosis by $\simeq 30\%$, a significant improvement. Due to these problems MATLAB changed its default colourmap from jet to a newly created one called parula that does not suffer from these pitfalls (Eddins 2017).

In light of these limitations MBE includes perceptually uniform, sequential, color-blind friendly colourmaps: viridis, magma, plasma, inferno (Garnier, scale), and scales) 2016) and colour-blind friendly diverging colourmaps: coolwarm, PRGn and seismic (Moreland 2009). Parula is not included due to copyright issues. Viridis is set as the default in MBE. Since many researchers are familiar with jet it is also included (White et al. 2011; Vanni and Timothy H. Murphy 2014; Hillman et al. 2007). We recommend researchers check their jet data using these colourmaps. For correlational visualizations that include negative and positive correlation using diverging colourmaps should be used if the focus is on the full range and contrasting negative with positive correlation. For visualizations that display a measure that is linear and never falls below zero such as standard deviation maps (see appendix A.19) we recommend using perceptually uniform sequential colourmaps.

Fig. 3.7 was generated through the same data and pipeline as Fig. 2.3, with the only difference being the use of the viridis colourmap instead of jet. In this example it is immediately apparent when comparing the two that in Fig. 3.7 the areas of high correlation drop off more gradually as one moves further away from the seed compared to Fig. 2.3. Fig. 2.3 appears to suggest a drastic change represented by a yellow band separating "high" from "low" correlation, which clearly demarcates the boundary between these regions. This "false clarity" in where one functional module might begin and another end is due to how narrow the yellow band is in jet. A gradual change in correlation values as one moves further from the seed would therefore see a sudden beginning and end of yellow, biasing the user into believing an abrupt change in correlation values occur at the yellow band. Colourmaps alone with appropriate SPC maps thus may not provide as clearly demarcated functional modules as desired. Consequently it is unsurprising that many are opting for data-driven techniques to find functional modules within their data (Vanni, Chan, et al. 2017) (See Sect. 4.2.2).

For reference, Fig. 3.8 is also provided where the same was done for the correlation matrix in Fig. 2.7

3.5.2 Internal Design

The recommended characteristics of internal design from Code Complete 2 (McConnell 2004) form the basis of MBE's internal design. Of these, particular attention was paid to loose coupling, re-usability and extensibility.

Loose coupling means designing to hold connections among different parts of a program to a minimum (McConnell 2004). This minimizes work during maintenance. Encapsulation and information hiding can be used to design classes with as few interconnections as possible. Loose coupling is most evident in MBE when considering the relationship between the utility classes in the util directory and the plugins one level above. Each plugin is encapsulated in having no dependence on any other plugin. This is true even for plugins that do functionally depend on one another. For instance, in order to use the SPC map plugin one must import ROIs via the import ROI coordinates plugin otherwise SPC maps simply cannot be plotted. The spatial filter plugin is currently used exclusively to aid in improving alignment results. Although functional dependencies exist between plugins, there are no internal code-based dependencies between them. This frees a programmer to code in a plugin (where the majority of coding takes place) without fear that their edits might affect other plugins. Instead plugins are dependent on the utility classes which are also encapsulated and hidden from a user who is only programming one plugin. If a programmer wants to add functionality from one of the utility scripts to their plugin they are free to do so and do not need to modify the code that makes up the utility script. They only need to understand the utility code's output and what input is required.

Encapsulation was not used just to contribute to loose coupling, but to safeguard important code from anticipated changes. Metadata and the location of image stacks used in a session in MBE are all saved to mbeproject.json. This JSON file therefore undergoes a high amount of change, essentially changing with every action the user makes within the application. The JSON file has a delicate structure that if disrupted may cause unexpected errors that are difficult to find the cause of as the JSON file is not code and therefore a debugger's traceback does not extend to it. This is especially concerning as the JSON file is absolutely essential to all modules that perform any data processing in MBE. And yet, changing the JSON file with a typo might not result in an error message. Due to the highly volatile nature of the JSON file all code that directly manipulates it are strictly encapsulated in the project_functions.py script. Functions here for saving particular attributes have been tested and the a programmer who wishes to save attributes to the JSON must thus use functions that are known to be safe to use in this utility script. This code reuse minimizes the chance of errors due to faulty JSON file manipulation being introduced.

The relationship between utility scripts in the util directory and plugin scripts is also applicable to re-usability. Re-usability means designing a system so that one module can be reused by other modules (McConnell 2004). Functions and classes from the utility scripts are used many times by all other scripts. Two primary cases of inheritance relationships are where virtually all plugins inherit the default classes in plugin.py. The Liskov Substitution Principle (Martin 2002) is applied here: derived classes are more specific versions of the base class. i.e. most plugins are more specific versions of a "default plugin" both conceptually

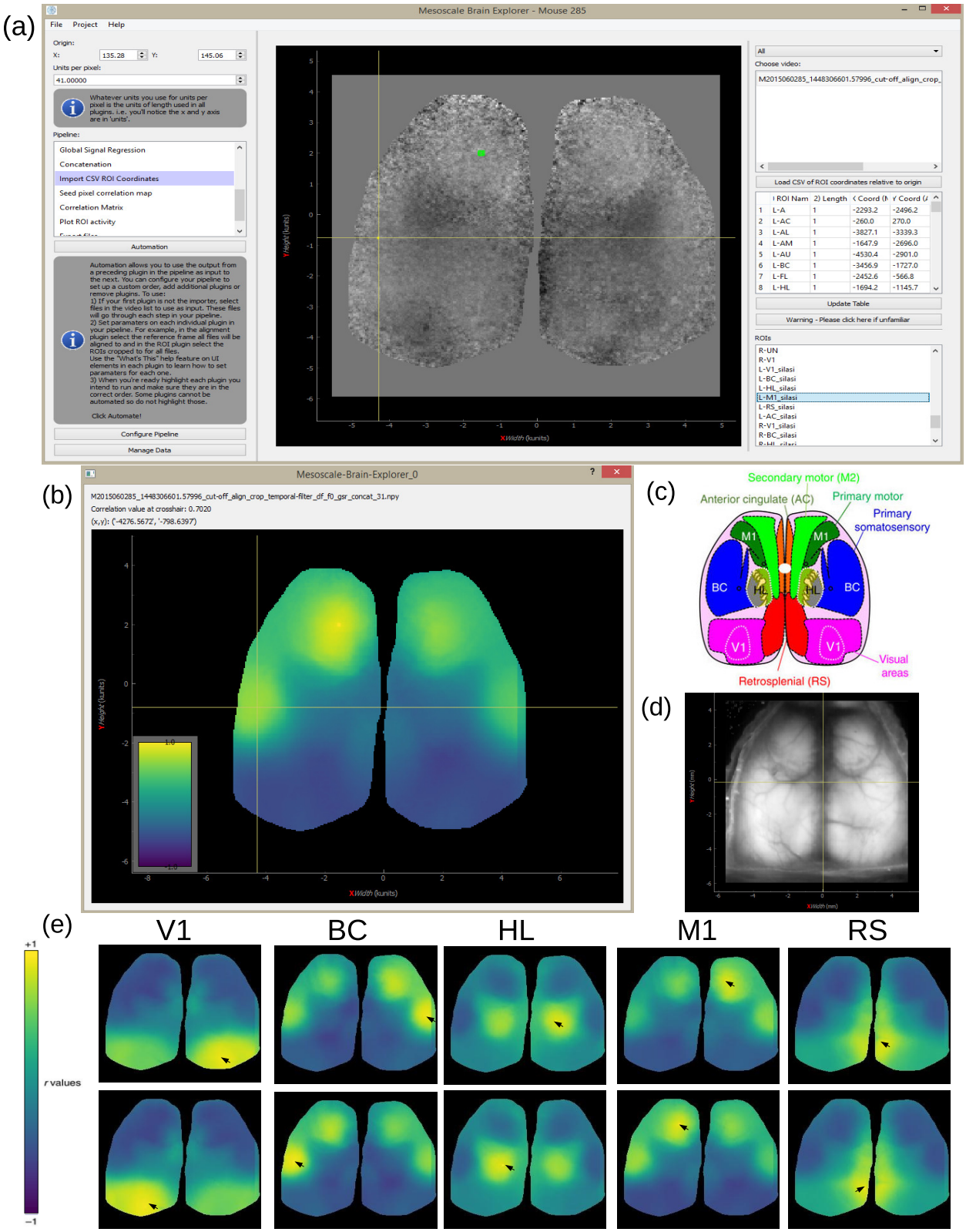
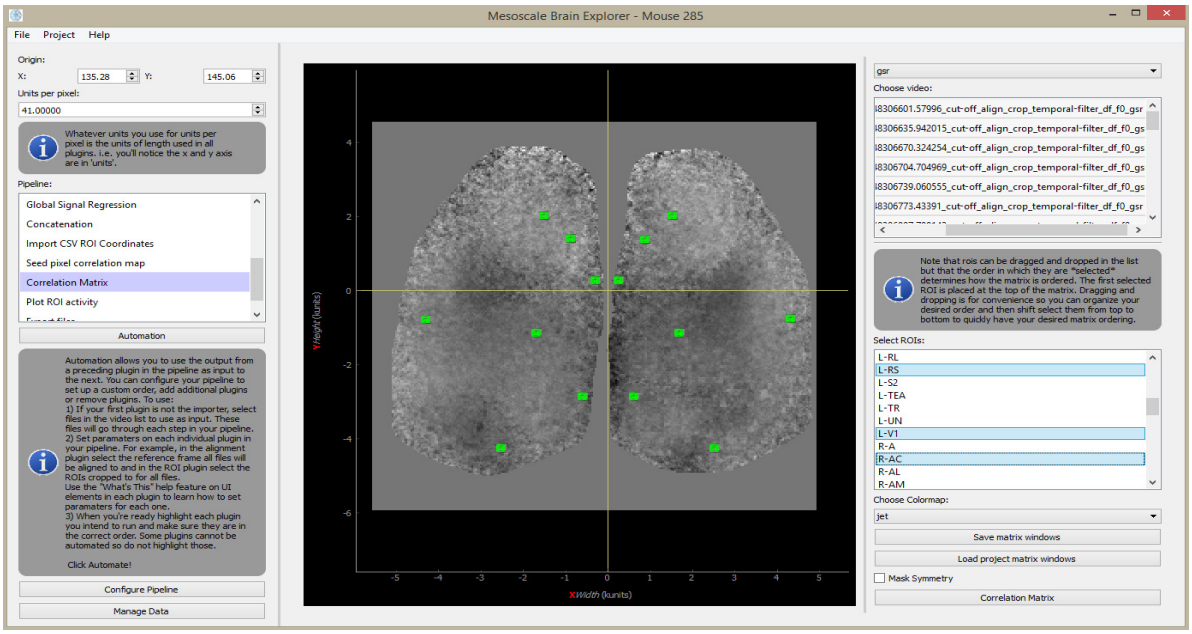


Figure 3.7: Fig. 2.3 repeated except with the viridis colourmap for comparison.

(a)



(b)

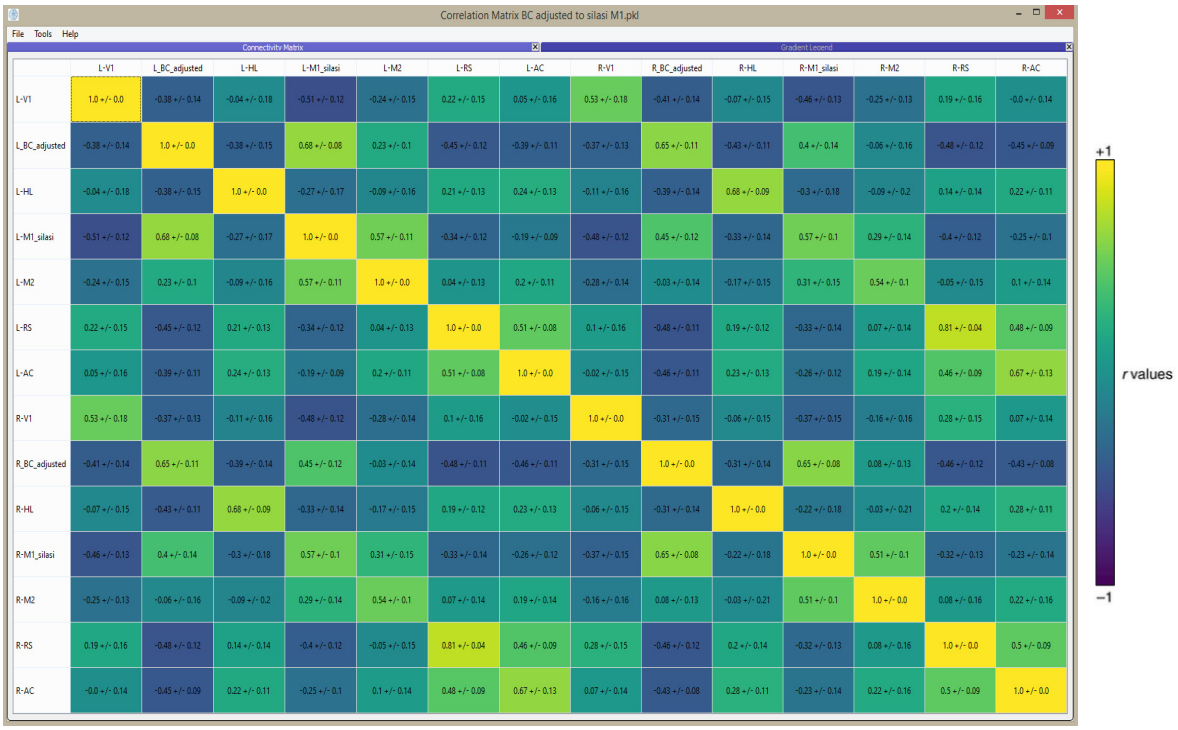


Figure 3.8: Fig. 2.7 repeated except with the viridis colourmap for comparison.

and functionally as most plugins make use of all UI components and functions provided by PluginDefault and WidgetDefault. Another example is where DockWindowSPC and DockWindowPlot classes that create the visualization windows for SPC maps and activity plots respectively both inherit DockWindow. Despite the amount of inheritance occurring overall (high amount of code re-use) inheritance does not occur many times for any one module. Therefore inheritance does not impede encapsulation to a significant degree (See Appendix 3.1.1 for an introduction to how inheritance can break encapsulation). During MBE's implementation it was a standard practice to refactor code written for plugins that was found to repeat itself across plugins and encapsulated it in its own script, utility function or class. This refactoring process has led to MBE having high code re-usability. This is in accordance with the DRY principle (See Appendix 3.1.1) thereby reducing information bloat, which is essential when implementing a system via the EVO model (Sect 3.1.2).

Polymorphic typing of plugin classes has contributed to both loose coupling and re-usability. During automation plugin classes selected to undergo automation are appended to a list in pipegui.py. Each plugin is treated as the same type where each one's check_ready_for_automation, get_input_paths and run functions are called. Conceptually all plugins are the same type, all being subclasses of the default plugin. However the calls to their functions are polymorphic in the sense that each function is realized differently depending on the plugin. This contributes to loose coupling as no additional module is required to transform plugins to a common type, thereby reducing the amount of system connections. This contributes to re-usability as the automation function can be applied to any newly added plugins (if implemented correctly). A programmer need not repeatedly implement code that handles how a plugin is automated and instead only needs to implement the specific characteristics of a plugin that define how it reacts to being automated. This adheres to the DRY principle, thereby reducing information bloat for more efficient implementation under the EVO model.

Extensibility means that a piece of a system can be changed without affecting other pieces (McConnell 2004). This is a key design feature of MBE as change is anticipated as different users will have needs that will require that they add their own code to MBE. The plugin architecture means that the addition or modification of plugins are the most likely changes that MBE will experience. Therefore, to abide by the principle of extensibility plugins were designed such that adding one or modifying one causes the least amount of trauma to the system. Designing MBE to anticipate such changes is thus key to ensuring long-term maintainability and usefulness. This principle is important enough to warrant its own tutorial showing how a plugin can be added to the system. This is covered in Appendix C.

Chapter 4

Discussion

The first research aim for MBE includes a design where a neurophotonics researcher inexperienced with programming can easily manage and share their own processing pipeline within the application. The second research aim includes a design where MBE is easy for programmers at the undergraduate level to modify and extend their own functionality onto without having to rewrite modules used often in the program. Both aims at their cores are questions of how computationally reproducible MBE is. Can it be shared and used by the non-programmer to reproduce a study and can it be shared and used by a programmer to modify the code so that a study can be reproduced? Below we assess MBE's strengths and weaknesses based on its reproducibility. We conclude with a discussion of MBE's limitations and further directions for development.

4.1 Strengths of Mesoscale Brain Explorer

4.1.1 Neuroscience

Y. Ma et al. observes that the advent of genetically encoded fluorophores that can report neural activity with high sensitivity, as well as modern technologies such as light emitting diodes and sensitive and high-speed digital cameras have driven renewed interest in wide-field optical mapping. However, with limited commercial platforms available, home-built implementations of rigs designed to image spontaneous activity have varied widely between groups. Analysis and interpretation of data have thus also been varied, leading to general confusion regarding what can be understood from the method in terms of sensitivity, quantitative accuracy, depth sensitivity and optimal implementations (Y. Ma et al. 2016). Generating reproducible research is thus made difficult because of the diversity of hardware and software in these workflows. As the data is moved between each program additional manual inspection, readjustment and perhaps combination with other data is required (Kitzes, Turek, and Deniz 2017). This problem is typically solved by custom expedient methods unique to each researcher or research group that make it difficult to capture the entire workflow to enable other researchers to reproduce a result (Kitzes, Turek, and Deniz 2017).

Replicability refers to the ability of a researcher to duplicate the results of a prior study if the same procedures are followed but new data are collected (Goodman, Fanelli, and Ioannidis 2016). This is distinct from reproducibility which refers to the ability of a researcher to duplicate the results of a prior study

using the same data as the original investigator (Goodman, Fanelli, and Ioannidis 2016). Replication is the foundation of cumulative science (Crocker and Cooper 2011) and yet replicability depends on a study being reproducible. This has been exemplified in studies showing difficulties with replicating published experimental results (Nekrutenko and Taylor 2012), largely due to absent experimental details required for reproduction (Ioannidis et al. 2009). This has led to a recent increase in retracted papers (Steen 2011) and higher numbers of failing clinical trials (Prinz, Schlange, and Asadullah 2011; Begley and Ellis 2012). This trend has since the early 2010s been referred to as the replication crisis. It is thus unsurprising that funding agencies, research institutes and publishers are all gradually developing policies to reduce the withholding of computer programs relating to research (Morin et al. 2012). Since October 2014, all Nature journals require a statements declaring whether the program underlying core results are available (Eglen et al. 2017). Since April 2015 Nature Biotechnology explicitly recommends referees give feedback on their ability to test code that accompanies a submitted manuscript ("Rebooting review" 2015). Also, since July 2015, BioMed Central adheres to newly introduced a minimum-standards-of-reporting checklist for BMC Neuroscience and several other journals, requiring that submissions include a code availability statement and for code to be cited using a DOI or similar unique identifier (Kenall et al. 2015).

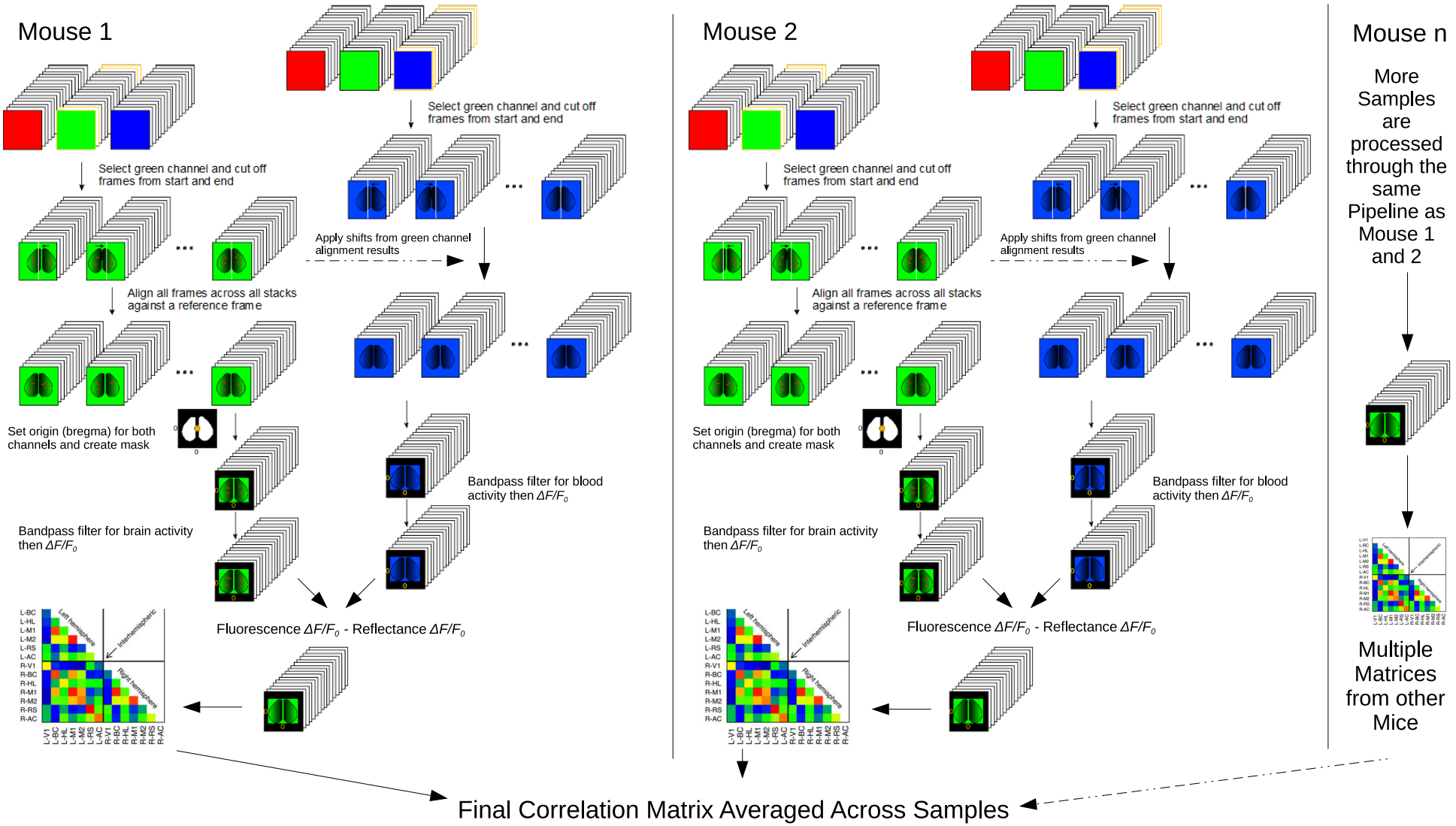


Figure 4.1: Pipeline used to create an averaged correlation matrix for an arbitrary number of mice.

MBE's offers users the flexibility to still set up custom expedient pipelines unique to a particular researcher without sacrificing reproducibility. As an example consider Fig. 4.1 which represents the pipeline used by MBE's Front-Line Analyst - primary user of MBE (FLA). This custom pipeline is noticeably more complex than the pipeline covered in Fig. 2.2. The FLA's study involves assessing the use of electroconvulsive therapy as a treatment for depression. Mice data are first recorded without any intervention (negative control). Mice then undergo a behavioural defeat paradigm as a model of depression and their brains are imaged to assess expected changes following depression (positive control). Finally, depressed mice are given electroconvulsive therapy sessions and subsequent brain activity recorded (experimental group). The FLA thus has three sets of data and she wants to assess changes in regional connectivity between these groups using a method that has a relatively easy neurophysiological interpretation such as through the use of correlation matrices. The final output of the FLA's pipeline in Fig. 4.1 is a single correlation matrix averaged across all correlation matrices outputted for each individual mouse in that set. Therefore this pipeline is repeated for three sets of data where in each set a correlation matrix is computed for each mouse and then all correlation matrices averaged.

As the same pipeline is repeated on each mouse, we can focus only on Mouse 1 in Fig. 4.1. The green channels for multiple brain recordings are imported for this mouse. Analogously to Fig. 2.2, image stacks are aligned to a reference frame, the location of bregma is chosen and set as the origin for the mouse, the hemispheres are cropped, a temporal filter is applied and finally $\Delta F/F_0$ is computed (No GSR is applied). However, instead of a correlation matrix being computed across all $\Delta F/F_0$ image stacks for Mouse 1, first hemodynamic correction is conducted by subtracting reflectance $\Delta F/F_0$ (blue channel) from the fluorescence $\Delta F/F_0$ for each $\Delta F/F_0$ image stack. The reflectance $\Delta F/F_0$ is processed through a parallel pipeline where the blue channel of all Mouse 1 image stacks are imported. However, instead of blue channel image stacks being aligned to the reference frames, each one is shifted based on the shifts found for each images stack's corresponding green channel image stack. This is to ensure that channels are aligned to one another following alignment. The user could also have opted to align the blue channel to a reference frame and then shift the green channel image stacks in accordance to the shifts found for the blue channel stacks. After these shifts the blue channel still lines up identically to its green channel counterpart and therefore the same hemisphere mask and bregma origin is applied to the blue channel, and can be done at the same time as when it is done for the green channel. Finally a temporal bandpass is applied, however this one is specific to the blue channel for parsing out the blood activity. Finally, reflectance $\Delta F/F_0$ is computed. After hemodynamic correction following fluorescence $\Delta F/F_0$ - reflectance $\Delta F/F_0$, we have multiple image stacks where a correlation matrix is computed for each and all matrices averaged to create a single matrix for Mouse 1. This is repeated for as many mice as the FLA desires or for however many mice there are in a particular group and then all averaged matrices for each mouse are themselves averaged to create an averaged matrix for the group of mice. Once this is repeated for another group of mice, correlation matrices representing entire groups can be compared and further statistical analysis performed (outside of MBE) on the matrix correlation values such as computing t-test statistics: effect size, sample size needed for particular power, confidence intervals, standard deviation, significance level and so forth.

Note that the creation of the reference frame in Fig. 4.1 occurs in two steps of its own. Once image stacks

are imported, an image stack and range of frames in this stack are selected and an unsharp filter is applied to these frames with a user-specified kernel size (See Appendix A.22) to enhance blood vessels which are the principal features aligned to. A user-defined percentage of the total width of an image stack from all sides of an image stack are then cropped to reduce noise from bone matter during alignment (ensures only blood vessels are used to ascertain best alignment). All frames are then averaged to create the reference frame. These steps are not shown in the figure. Note also that these steps occur to the data to be aligned as well. Alignment takes place for the unsharp filtered and cropped data and the best shift discovered is then applied to the unaltered data. These advanced alignment options were applied by the FLA and were not available when analysis was performed to create the pipeline in Fig. 2.2.

The automation of high-throughput processing pipelines through standardized methods has an ethical dimension. A study that achieves only 80% power still presents a 20% possibility that the animals have been sacrificed without the study detecting the underlying true effect. It has been observed that the average power in neuroscience animal model studies may be between 20-30% (Button et al. 2013) - suggesting that underpowered studies that waste animals appear to be the norm. There is ongoing debate regarding the appropriate balance to strike between using as few animals as possible in experiments and the need to obtain robust, reliable findings. Standardized high-throughput processing has become a means of ensuring that the standards of quality and robustness of results after processing is high for each animal modelled. This ensures that achieving a high enough sample size for proper statistical power is more achievable, thereby reducing the amount of animals that are wasted.

MBE is a user-friendly flexible open-source Python-based image analysis and visualization tool that standardizes and automates common processing steps and analysis in mesoscale wide-field optical mapping to enhance reproducibility of mesoscale brain mapping pipelines, thereby providing a framework for more reproducible research when producing correlation matrices, SPC maps and brain activity plots or exploring different outcomes based on changing parameters in a processing pipeline. These tools allow for robust and easy-to-understand exploratory analysis of spontaneous mouse brain activity, where the connectivity strength and patterns among brain regions can be inferred from correlation strength. Accordingly, MBE has thus far (as of August 2017) been downloaded 60 times and is known to be regularly used by two researchers in our lab, one who has used it extensively for seven months and another for two months.

4.1.2 Software Engineering

A recently published book by Kitzes, Turek, and Deniz (2017) reviews the current state of data-intensive science and provides 31 case studies of software across various fields used for reproducible research workflows. They identify six core recommendations repeated by scientists, regardless of field, for improving software reproducibility. An additional seven recommendations are also made. Of these, we identify the following that are relevant to MBE:

Code Sharing

Five recommendations involve code sharing that include 1) Version control your code, 2) Host it on a collaborative platform such as GitHub, 3) Open your data, 4) Use free and open tools and 5) Get a Document

Object Identifier (DOI) for your data and code.

All five of these recommendations are easily met in MBE's development by using version control through the widely respected (Eglen et al. 2017; Sandve et al. 2013; Eglen et al. 2017) collaborative platform GitHub. Exact reproduction of results is often tied to a specific version of code. Therefore, without systematically archiving code through version control code cannot be backtracked and this can cast doubt on previous results (Sandve et al. 2013). Allowing open source access to code for scrutiny and extension is widely held (Eglen et al. 2017; Sandve et al. 2013; Eglen et al. 2017; Halchenko and Hanke 2015; Stodden and Miguez 2014) as a strong step towards improving reproducibility. MBE and all tools and sub-modules MBE uses have licences that guarantee code is open and free. Any user can thus see any aspect of MBE's code and modify it for their own purposes without restriction as all code is hosted on a public repository. Finally, as DOI's are the backbone of academic reference and metric systems (Kitzes, Turek, and Deniz 2017; *Making Your Code Citable · GitHub Guides* 2017) MBE's GitHub repository is integrated with Zenodo (*Making Your Code Citable · GitHub Guides* 2017) which will assign it a citable DOI once version 1.0.0 (See Appendix B) is released.

GitHub hosting further offers user support. Establishing a user community is a suggested means of handling the burden of feature requests and for tracking bugs (Gorgolewski and Poldrack 2016). This can easily be dealt with by asking issues be posted on a GitHub repository (Eglen et al. 2017) which is precisely what MBE does.

Avoid excessive dependencies and when dependencies cant be avoided, package their installation

Often the first obstacle to for use, sharing, and adoption of any software is the battle to get it working on a different machine than that on which is was created (Kitzes, Turek, and Deniz 2017). Installation of dependencies is a critical stumbling block to sharing and extending academic code (Kitzes, Turek, and Deniz 2017). Kitzes, Turek, and Deniz (2017) note that their case study authors often used lightweight strategies that were often fragile to cross-platform-configuration issues (e.g. bash scripts and makefiles) whereas more robust solutions such as virtual machines were more clunky and often less transparent. While older Window's systems will require at least 2008 Visual C++ Redistributable (installation instructions included on MBE's GitHub repository (Haupt 2017)), installing MBE on newer Window systems (7, 8.1, 10) is as simple as downloading and running an executable. This portability makes sharing MBE very straightforward on Windows systems. Installing MBE on Linux systems requires installing dependencies with instructions provided on MBE's GitHub repository and then running the Python file. This is admittedly a far more complicated process, but all required dependencies have Python packages. Of these, only OpenCV - a very popular Python library - may require more than a single step to install correctly. Moreover, once dependencies are set up in Linux to run MBE, MBE's code can be modified without any additional steps.

Document all operations that occur on data and files

This recommendation is key for any research to be reproducible as the full sequence of pre- and post-processing steps are often critical in order to reach a researcher's achieved result (Eglen et al. 2017; Sandve et al. 2013; Eglen et al. 2017; Halchenko and Hanke 2015; Stodden and Miguez 2014). MBE's project-

focused architecture helps a user document their processes, however the user is still responsible for their own documentation to a degree. Any session of use in MBE must take place in the context of a project where MBE saves all project parameters including parameters set by all plugins and saves the user's pipeline order (See Chap. 3). This collectively comprises the user's pipeline configuration. As a user processes data in their pipeline the name of a process an image stack has gone through is appended to the image stack's name and all intermediate results in a pipeline are saved.

All intermediate data are saved in a standardized format in MBE - typically numPy arrays (Walt, Colbert, and Varoquaux 2011) for image stacks though any data used to generate intermediate numPy arrays (e.g. ROIs) is also saved. Having easily accessible intermediate results in a standardized format can often reveal discrepancies, bugs or faulty interpretations with greater ease and track them to specific steps. Consequences of alternative programs and parameter choices at individual steps can be revealed directly. And crucially, for user-friendliness, this allows parts of the pipeline to be rerun when the full pipeline is not readily executable (Sandve et al. 2013). Saving intermediates in a standardized form thus does not only aid to document all operations that occur by also recording outputs but also aids in code maintainability and application user-friendliness, both which also aid in enhancing reproducibility.

In addition to automatically outputting intermediate results in a standardized format any data comprising the end-result visualizations such as seed-pixel correlation maps, correlation matrices or activity plots are all automatically saved to file - again to the standardized NumPy array, but CSV files and jpeg images of the same data is also outputted where appropriate. Besides backing up the crucial end-result data automatically, this allows the raw numbers of any output to be consulted if needed later - a level of scrutiny that serves to enhance reproducibility (Sandve et al. 2013). The visualization windows introduced in chapter 3 offer another function. With these, one can simply modify the plotting procedure, instead of having to redo the entire analysis, if a variation of a plot is required on the same data. Besides improving reproducibility (Sandve et al. 2013) this allows for excellent exploratory analysis.

In addition to an image stack's name being modified, a list of manipulations any image stack has gone through can also be viewed within MBE for any data item that has undergone processing. As of version 0.8.0 (See Appendix B) however this does have limitations. For one, data processed through the same set of plugins where one has different parameters (e.g. temporal filtering with two different bandpasses) will not be differentiated in MBE. The user is therefore responsible for documenting this themselves. However, manual documentation can then more easily get out of sync with how the analysis was really performed for the final result as the user may process different image stacks in myriad ways in the same project for exploratory analysis (Sandve et al. 2013). The full analysis workflow is better stored in a form that allows for direct execution (e.g. makefiles or shell scripts (Schwab, Karrenbach, and Claerbout 2000; Heroux and Willenbring 2009; Sandve et al. 2013)) so that analysis can reproduced in an automated way so as to avoid human error. This limitation is known and MBE's design - especially the design of the interface of the Manage Data window (See Chap. 3) - is built with the justified (Sandve et al. 2013) assumption that documentation of operations will include layers of increasingly detailed metadata for any data item in a project. Moreover, if the user creates a pipeline configuration, sets the parameters for each plugin and doesn't change them again after data processing, the user can look at their processed data and exactly what

processes they went through. This allows later reproduction of their findings if the same project is loaded. More importantly, the pipeline configuration of a project is saved to a small JSON file that is easily shareable. Any other user can then load this file in their copy of MBE to replicate the pipeline on their own data with ease.

At a minimum - as recommended by Kitzes, Turek, and Deniz (2017) - a user of MBE should be able to have sufficient details on parameters recorded along with easily recorded manual procedures allow MBE users, in a year or so, to reproduce their results. This would mean their result should be reproducible for others as well if appropriate instructions are given.

Automate everywhere possible

Whenever possible, it is recommended to rely on automated execution of processing steps instead of manual procedures to modify data. Manual procedures are not only more inefficient and error-prone, they are also more difficult to reproduce (Sandve et al. 2013). The reproducibility of a project can thus greatly be enhanced through the creation of a single script that automatically executes all stages of a processing pipeline (Kitzes, Turek, and Deniz 2017). Although in some instances such automation may be unrealistic or impossible due to project constraints, in which case detailed documentation of all non-automated steps should be created (Kitzes, Turek, and Deniz 2017). This is exactly what MBE achieves as plugins that can be automated are automated allowing segments of a pipeline to be automated. Moreover, MBE's pipeline configuration allows for duplicate entries each with their own parameters as well as multiple pipelines processing simultaneously with multiple entry points with different data. This means that the processing of data from multiple datasets, even when those datasets each need to be processed the same way but with different parameters, can all be automated at once. i.e. The automated processing of one dataset does not have to be finished and then parameters changed before the automated processing of the next dataset can be performed. All pipelines can be processed at the same time. This mitigates difficulties where a single processing step may require *ad hoc* analysis to determine what the best parameters are for a single plugin. Multiple pipelines can be set up such that this plugin is tested automatically with many parameters. The processing can commence and the user can check to see which output provides the most promising output at the end. Above MBE meets five of Kitzes, Turek, and Deniz (2017)'s six core recommendations and further meets four of their further recommendations.

Ease of Use

A final concern raised by Kitzes, Turek, and Deniz (2017) is that their case study authors reported that the bottleneck to adopting practices was related to a diversity of skills (Kitzes, Turek, and Deniz 2017). It was universally reported from various researchers that their work might have been more efficient and reproducible if their team were all more familiar with the software tools they were trying to make use of. Case studies clearly showed researchers thought collaborators unfamiliar with tools used by their colleagues often crippled their research (Kitzes, Turek, and Deniz 2017). Scientists however were often unwilling to disenfranchise their collaborators on ethical grounds and would thus elect to use more widely used tools, accepting the frustration with inefficiency as the price of collaboration (Kitzes, Turek, and Deniz 2017).

This however also affects reproducibility as the tools selected were often tools such as easy-to-install point-and-click Microsoft Word, Excel, or MATLAB which are noted as particularly problematic fallbacks, as their closed-source GUI-based nature is fundamentally fragile to reproducibility issues (Kitzes, Turek, and Deniz 2017).

MBE is a workable solution to both concerns. It has a GUI-based architecture that is familiar to most computer users and therefore most researchers. A tutorial for using it further exists online and the application is full of contextual help popups that further aid the researcher. However as an easy-to-install point-and-click solution it does not suffer from reproducibility issue that others do as is evidenced by how strongly MBE meets many recommendations listed above. User interaction is automatically documented, most processes can be linked in a pipeline and automated, and importantly, all code is open-source.

Moreover, MBE's plugin architecture where pipeline processing steps are encapsulated into plugins make it easy to extend or modify MBE's functionality, which can be done via the Python programming language - one of the most widely used languages in academic research. This makes it a fundamentally different solution from for example Microsoft Excel's code if it was open source as even an experienced programmer might find it difficult to modify.

4.1.3 Summary

In short, the external design discussed in Chap. 3 highlights how its ease of use adds to MBE's reproducibility and transferability whereas the internal design also discussed in Chap. 3 highlights how the use of software abstraction principles similarly makes MBE easy to modify (which aids in reproducibility) or extend MBE. This extension architecture has already been put to use as new plugins were introduced with relative ease that could solve individual user's issues such as dividing fluorescence dF/F0 by reflectance dF/F0 which can be used for hemodynamic correction (Ron D. Frostig and Chen-Bee 2009; Y. Ma et al. 2016; Xiao et al. 2017).

4.2 Weaknesses of Mesoscale Brain Explorer

Before discussing MBE's weaknesses we must first differentiate between weaknesses that are limitations only as features that are missing and weaknesses that are the result of poor design choices. We cannot possibly cover all missing features as MBE was built for the work done in one lab, but we can highlight particular features that *should* eventually be part of MBE as they are relevant to work done in our lab. These weaknesses can conceptually be divided into software engineering weaknesses (poor design choices) and neuroscience weaknesses (limitations and future directions).

4.2.1 Poor Design Choices

Testing

Most software has bugs (Kitzes, Turek, and Deniz 2017). A study by *Coverity scan open source report 2014* (2015) found 0.61 errors per 1,000 lines of code of source code in open-source projects and 0.76

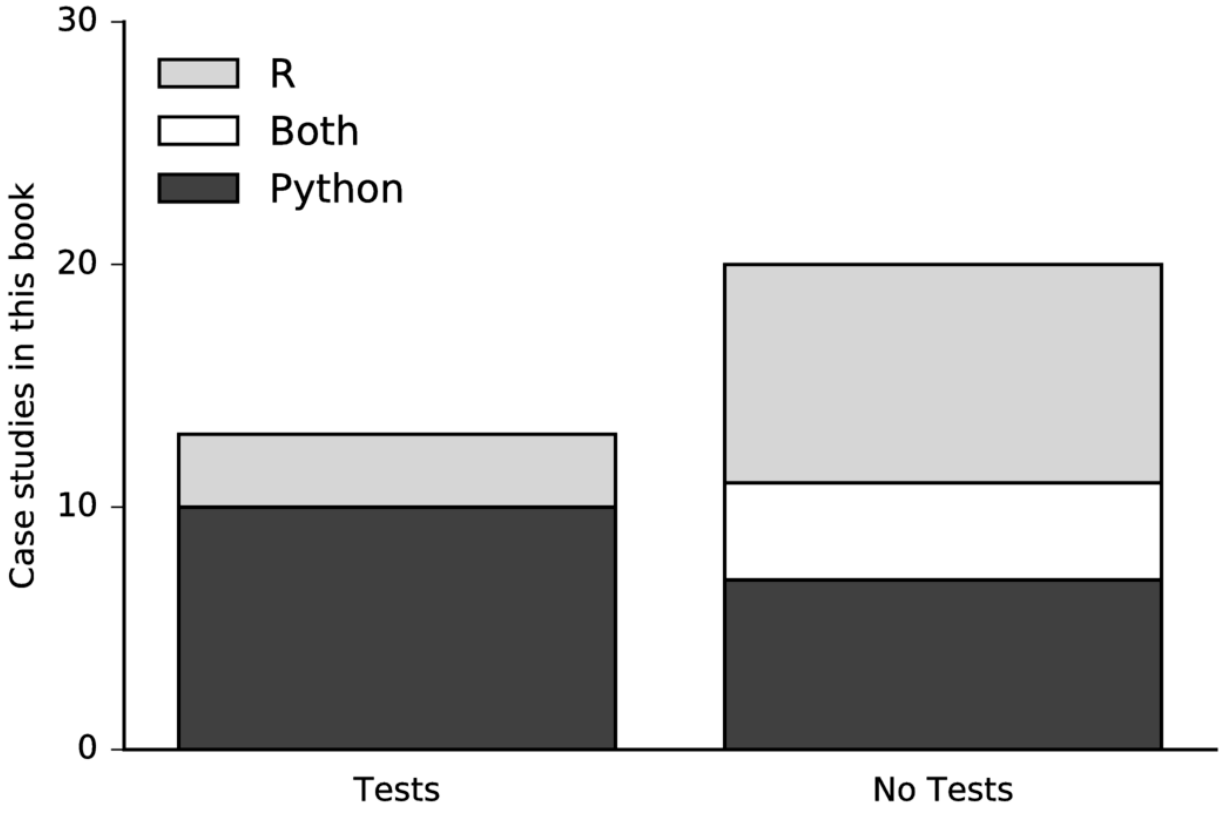


Figure 4.2: Prevalence of testing in code by language used in out of 31 case studies. In: J. Kitzes, D. Turek, and F. Deniz (2017). *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Science*. Oakland, CA: University of California Press. URL: https://www.gitbook.com/book/bids/the-practice-of-reproducible-research/details (visited on 06/26/2017). Figure copied with permission from the University of California Press.

errors per 1,000 lines of code in commercial software. Scientific software is no exception. A study by Reinhart and Rogoff (2010) was used to justify economic austerity measures in Southern Europe. However, a later study found that errors in their Excel spreadsheet led to the wrong conclusion (Herndon, Ash, and Pollin 2014). If their code has been scripted with sound testing practices, their errors might have been avoided, discovered, or corrected before harm was done (Kitzes, Turek, and Deniz 2017). Code testing is a critical step in the software industry, yet the practice is not yet widely adopted by researchers both inside and outside neuroscience (Kitzes, Turek, and Deniz 2017; Eglen et al. 2017; Wilson et al. 2014; Axelrod 2014; Merali 2010; Soergel 2015). Indeed, Kitzes, Turek, and Deniz (2017) find, based on their 31 case studies across various fields, that most researchers did not include any tests in their code (Fig. 4.2). They acknowledge a need to understand why the incentive of long-term efficacy does not lead more researchers to test their code as a standard practice (Kitzes, Turek, and Deniz 2017). Given the consensus that code testing greatly enhances code reproducibility (Kitzes, Turek, and Deniz 2017; Eglen et al. 2017; Wilson et al. 2014; Axelrod 2014; Merali 2010; Soergel 2015) it is unsurprising that Kitzes, Turek, and Deniz (2017) lists "Test Everything" as one of their six *core* recommendations, rather than merely an additional one.

Despite this, MBE, as of 0.8.0 (See Appendix B), has no code testing implemented. The problems this lack of testing causes are already evident as our research has often been stalled by small typo errors that make their way into released versions. As noted in Fig. 4.2 this does not make MBE an outlier when considering academic software, but it does arguably make it an outlier when considering academic software written in Python. The reason for this is that there are many popular testing frameworks available in Python (such as nose or unittest (Kitzes, Turek, and Deniz 2017)) with active online communities and documentation that can aid development. As such, this is easily the single greatest inexcusable weakness in MBE's design and must be amended in further versions.

Various tests can be implemented into MBE. At the low level individual functions should be tested for robust handling of a range of inputs expected of them (called unit testing). Since functions within the plugin classes themselves evolve over time these should have unit tests to ensure that the change still handles all cases expected of a particular function. At a high level MBE as a whole can also be tested for returning correct answers on simulated data (this is called system testing) (Wilson et al. 2014). For a program with a GUI such as MBE, automated UI tests should also be implemented using simulated user interaction to ensure that through the full range of possible user interaction, no user interaction easily breaks the program. Once these test suites are implemented tests could be automatically run after any modification to MBE's source code to ensure the change doesn't break any test.

For MBE where most processes are computationally expensive on large datasets, running these tests on each modification will be time-consuming and impractical. Continuous integration is a means of outsourcing this task to a server where code is automatically and continuously tested, allowing researchers to focus on implementation without worrying about constantly running their test suites. If a bug is found, a continuous integration server will send out a notify the programmer. As an industry standard practice many services such as Travis-CI or Jenkins offer free continuous integration servers that can be made to work with Python test suites (Eglen et al. 2017; Kitzes, Turek, and Deniz 2017). Of the few researchers in Kitzes, Turek, and Deniz (2017) case studies that used continuous integration, its use was lauded as essential. Reproducible practices are easiest to adopt when they save time. Continuous integration is just such a practice and along with automated UI testing, unit tests and system testing should be part of MBE's future design to ensure long-term efficacy.

Transferability

The measure of MBE's success is not that a different software engineer would design the same system, it is that the particular researchers it is designed for find it useful (Sedlmair, Meyer, and Munzner 2012). This is referred to as transferability and takes precedence as meeting the needs of the end-users requires intrinsically subjective field work (B. Brown, Reeves, and Sherwood 2011). This brings with it possible poorer design however. As an example consider the alignment plugin. The UI for this plugin is notably complex and version updates iteratively updated these components based on user feedback. In software engineering, the open/closed principle states that software entities (classes, modules, functions etc.) should be open for extension, but closed for modification (Martin 2002). This principle exists to encapsulate code that undergoes a high degree of change and separate it from other code that should not be changed, thereby minimizing

bugs. This principle is abided by in the import plugin. Here the fileconverter.py script and the file fileimporter.py script are separated. This is by design so as to encapsulate the expected need for supporting further data types for import. Programmers can modify fileconverter which is expected to undergo many changes by adding new file conversion functions. Since changes are made only in this script, no changes are made to fileimporter.py, minimizing the potential for bugs. i.e. fileimporter.py is closed for modification (when modifying only the fileconverter script) and open for the extension of supporting additional data types via fileconverter.py. This "extensible layer" is not provided in all plugins, despite plugins like alignment.py undergoing a high degree of change. This is largely because the principle of transferability supersedes concerns over violating the open/closed principle. Once functionality is implemented and user feedback is positive, no additional modifications are made. In the case of the alignment plugin there doesn't seem to be a clear way to create an extensibility layer since what functionality might be extended is unclear. Modifications made to the plugin have added UI components to re-organize what control the user has over functionality the plugin already provides. No new "alignment algorithm options" have been added. Conceptually, the alignment plugin is simply thought of as an incomplete singular plugin that won't need any extending once complete.

Two other examples of transferability arguably leads to bad design from a software engineering perspective is the saving and loading of NumPy arrays (.npy) to file across all of MBE's functionality and the lack of standardized comments within the code. Using a standard format across all plugins is a strength. However, once the standard is set it is difficult to change as all plugins would need to potentially be redesigned. HDF5 is arguably a much better standard format for MBE as it is a format specifically designed for handling massive datasets which MBE often must handle (Collette et al. 2017). The NumPy format was however chosen simply because our lab's users are more familiar with manipulating it manually. Likewise with undocumented code, the template plugin that is expected to see use by programmers is heavily documented. However, most other code isn't simply because we do not expect members of our lab (the primary target end-user of MBE) to modify these sections. Code documentation thus remains a low-priority "to-do" rather than a requirement as it is in software engineering.

4.2.2 Limitations and Future Directions

MBE, as of version 0.8.0 (See Appendix B), only handles model-based bivariate unimodal undirected functional connectivity analysis via Pearson correlation coefficients. Causal inference, directionality and lags are therefore not considered and MBE cannot be used to attain a full picture of mesoscale connectivity.

Model-based vs Data-driven Connectivity Methods

Model-based methods rely on prior knowledge of spatial or temporal patterns (K. Li et al. 2009). ROIs are selected as "seeds" and temporal frequency ranges are selected as bandpasses and the correlations between seeds and other regions are used to generate maps of connectivity. Many processing steps such as alignment rely on input parameters from the user. Selecting the seeds, temporal filter and parameters of various plugins typically requires strong prior neuroscience knowledge or experience (K. Li et al. 2009). The Allen Institute's mouse brain connectivity atlas (Oh et al. 2014), which contains ROI coordinates of known anatomical

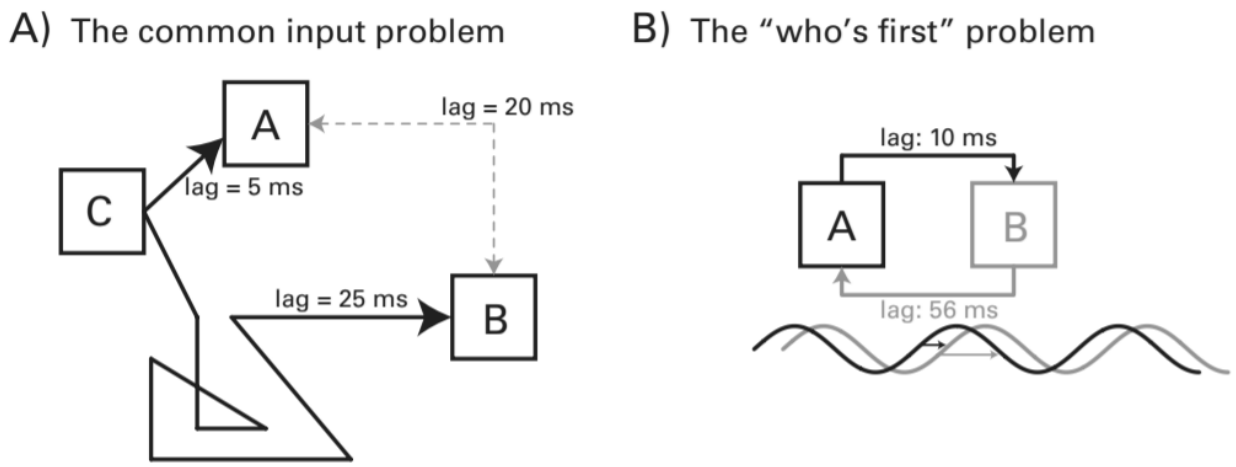


Figure 4.3: Two common problems when interpreting temporally lagged bivariate connectivity. Figure copied with permission from The MIT Press. Mike X. Cohen (2014). *Analyzing Neural Time Series Data: Theory and Practice*. English. 1 edition. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-01987-3.

locations, is used for analysis covered in Chap. 2 (Haupt et al. 2017). If coordinates for a sensory stimulus are not available experiments are conducted where mice are exposed to the stimulus. Sensory-evoked average maps are generated from the data to find the most activated region, on average, based on particular sensory input. Model-based methods are widely used for functional connectivity analysis as such techniques are easy to implement and physiological interpretation is easier (K. Li et al. 2009; Johansen-Berg et al. 2004; M. D. Greicius et al. 2004; Fox, Snyder, et al. 2005; Xiong et al. 1999; Lim, J. LeDue, et al. 2013; Silasi, Xiao, et al. 2016; Vanni and Timothy H. Murphy 2014; Xie et al. 2016; Chan et al. 2015). However, despite their popularity, model-based techniques are ill-suited for uncovering connections where no prior knowledge exists. The requirement for prior knowledge constrains the exploration of possible functional connectivity (L. Ma et al. 2007).

As an example consider cross-frequency coupling which refers to the statistical relationship between brain activity in two different frequency bands (M. X. Cohen 2014). Moreover, there are theories proposing a key role of cross-frequency coupling in information processing in the brain (Axmacher et al. 2010). Assessing the best bands for two ROIs being compared creates a potentially huge search space as all frequencies for each must be considered. Without a workable hypothesis based on prior neuroscience knowledge on which frequency bands to consider, finding two frequency bands that are coupled becomes a very time-consuming and daunting task. And yet, MBE only offers support for manual frequency band selection with a power spectrum transform that might be used to assess a frequency band's suitability. MBE would ideally include a tool that searches for frequency bands with high coupling or at the least include an option making it easier for a researcher to consider and check many frequency bands at a time rather than one by one.

A full exploration of brain connectivity would thus require what are called data-driven methods where no prior information about the spatial or temporal pattern of connectivity is known (K. Li et al. 2009).

Data-driven methods include decomposition techniques such as Principal Component Analysis (PCA) or Independent Component Analysis (ICA) and clustering techniques such as hierarchical clustering analysis (K. Li et al. 2009).

PCA represents observed brain activity time course with a linear combination of orthogonal contributors (K. Li et al. 2009). Each contributor is made of a principal component (a temporal pattern) multiplied with an eigen map (a spatial pattern). PCA is an explorative technique and thus helps to explore functional connectivity of the whole brain being imaged (K. Li et al. 2009). The linear combination of basis vectors PCA identifies highlights global spatial features of brain activity by identifying patterns of large-scale covariance (M. X. Cohen 2014). PCA however has several limitations (K. Li et al. 2009; Baumgartner et al. 2000) and is therefore more commonly used as a pre-processing dimensionality reduction step for further analysis such as ICA and various Blind Source Separation techniques (X. Chen, Z. J. Wang, and M. McKeown 2016).

ICA, like PCA, seeks to find a linear combination of components. However, ICA searches for components in data that are as independent as possible rather than orthogonal (Hyvärinen and Oja 2000). There are two common ICA algorithms, each with its strengths. The "Fixed-Point" algorithm outperforms the "Infomax" algorithm in terms of spatial and temporal accuracy whereas Infomax is better in global model estimation and noise reduction (Esposito et al. 2002). Data can be decomposed into spatially independent components and spatially independent time course (sICA), or temporarily independent components and temporarily independent time course (tICA). A researcher may want one or the other depending on the characteristics of the signals to be estimated (M. J. McKeown et al. 1998; V. D. Calhoun et al. 2001). A major problem with ICA however is that it is a noise-free generative model, meaning that the linear combination decomposed by ICA will include noise as part of its framework instead of identifying noise in data and extracting it (K. Li et al. 2009). Probabilistic ICA was developed to solve this problem which corrupts data intentionally with additive Gaussian noise (Beckmann and S. M. Smith 2004).

Clustering techniques partition data into different clusters based on some distance metric, usually the intensity proximity of the time course or more commonly in functional connectivity analysis the similarity between time courses (K. Li et al. 2009; Golay et al. 1998). Distance measurements are often derived from Pearson correlation coefficients (K. Li et al. 2009). Various clustering algorithms exist. Popular algorithms include fuzzy clustering, hierarchical clustering and k-means clustering (Hartigan 1975; Windischberger et al. 2003; S et al. 2012).

Given our lab has made use of PCA for mesoscale connectivity analysis and that we are actively exploring the use of clustering analysis and ICA it seems appropriate that these methods will eventually be implemented in MBE. Presently PCA would be designed as a plugin with automation functionality so that it can easily be coupled for preprocessing to ICA or other plugins that might require its output. ICA and clustering plugins can be developed as the need arises. There is concern here over transferability emphasis leading to undue violation of the open/closed principle (see sect. 4.2.1). As we've noticed there are many ICA algorithms and therefore whether these should all be within one plugin with an extensibility layer or all be separate plugins can only be answered once the design phase for these plugins is entered.

Bivariate vs Multivariate Analysis

Most but not all brain connectivity measures are bivariate, meaning that they involve interactions between only two brain regions (M. X. Cohen 2014). Some brain connectivity measures, such as correlation matrices (see Chap. 2, Sect. 2.5.2) may initially seem multivariate (one-to-all or all-to-all connectivity) but are in fact mass-bivariate measures because each step of the analysis involves connectivity between only two regions. Although this simplifies analysis, bivariate analysis can inflate or misrepresent estimates of relationships if the underlying network structure is actually multivariate. This is therefore a concern for mesoscale brain mapping as the brain is a highly multivariate system (M. X. Cohen 2014). Consider the case when region C is causally linked to both regions A and B, but A and B are not causally linked. Bivariate assessment will inflate the likelihood of reporting connectivity between A and B as region C is not considered. This is especially a concern for mapping connectivity via spontaneous brain activity. For sensory-evoked activity mapping this inflation can at least be partially mitigated due to controlled condition comparisons of connectivity. Partial correlations are another means to offset this concern. Partial correlations allow the user to measure a linear or monotonic (via Pearson or Spearman correlation coefficients) relationship between two variables A and B while holding a third, C, constant (M. X. Cohen 2014). This removes shared variance between C and A or C and B, reducing the correlation mapped between A and B when C is causally linked to either or both A and B (M. X. Cohen 2014). Partial correlation is also multivariate beyond three ROIs as extensions of the method allow the user to hold constant more than one variable. However, partial correlation is a model-based method and therefore suffers from the same limitations. ROIs that will be held constant need to be selected by the user based on prior neuroscience knowledge. Another concern is that activity in one region that is 0ms, 10ms or 100ms lagged from another can all be synchronized equally strongly (M. X. Cohen 2014). However, commonly used bivariate measures used to infer connectivity (e.g. correlation coefficients) do not explicitly capture this lag and may therefore report low correlations when the activity between two regions is highly synchronized. Cross-correlation Analysis (CCA) could be used to find a maximal correlation between activity A and B for a particular range of lags introduced between the two. Both partial correlation and CCA are straightforward to implement and will likely be introduced in future updates to address these limitations.

All data-driven methods mentioned however are multivariate and do not suffer from either limitation. We have recently explored clustering methods applied to mesoscale functional mapping of the mouse cortex. Since data-driven methods consider all inputted data it is unsurprising that we have found that mesoscale mapping of whole-brain recordings have revealed distinct macroscale functional modules (Vanni, Chan, et al. 2017). We are actively exploring decomposition methods in the same vein. Ultimately however, bivariate methods are easier to understand, implement, visualize, and statistically quantify (M. X. Cohen 2014). They are also often prerequisites for more complicated multivariate analysis. ICA for instance often uses iterative methods based on minimizing Mutual Information (MI), a bivariate measure.

Multimodal Analysis

The data-driven multivariate methods mentioned do not meaningfully integrate data from different brain monitoring modalities. For example, electroencephalogram recordings cannot simply be combined with GCaMP6 time-course activity in a single model on which clustering or decomposition analysis is performed.

When bivariate connectivity analysis is considered only two specific brain activity indicators are considered based on neuroscience knowledge. In this case it is easier to constrain regions and the modalities used such that cross-modality connectivity analysis can be meaningfully performed using a specific hypothesis and a neurophysiological interpretation reached (Chan et al. 2015). It is anticipated however that synergistically combining neuroimaging techniques will provide an unprecedented opportunity for understanding brain function as each brain spontaneous activity measuring techniques (Vesicle release indicators, neurotransmitter indicators, voltage indicators, calcium indicators, intrinsic signals) is an indirect reflection of the underlying neural activity at a specific spatiotemporal scale and thus provides a different aspect of brain function (Biessmann et al. 2011). In recent years and in the fMRI literature multivariate Joint Blind Source Separation (JBSS) methods have emerged to meaningfully integrate data from brain monitoring modalities (Vince D. Calhoun, J. Liu, and Adali 2009; Correa et al. 2010; X. Chen, Z. J. Wang, and M. McKeown 2016). A further description of these methods is outside the scope of this paper. These methods are not well established as techniques for functional mesoscale mapping of spontaneous mouse brain activity. Even in the FMRI literature they remain poorly understood as, to date, no single study has investigated all possible JBSS methods and demonstrated their strength and weakness thoroughly for any one specific neurophysiological problem (X. Chen, Z. J. Wang, and M. McKeown 2016).

JBSS methods are complicated and should only be implemented once their utility is fully understood. Moreover, even if their implementation is eventually to be realized, MBE must first incorporate other methods. ICA for instance is a preprocessing step for various JBSS methods. To implement ICA however, MI first needs to be implemented.

At a simpler level, multimodal analysis is possible for model-based methods so long as method data can be imported into MBE as image stacks. e.g. vascular dynamics (intrinsic imaging) are often analyzed alongside neural activity to assess links between the two (Vanni and Timothy H. Murphy 2014). This is easily done in MBE. Mapping connectivity between sub-cortical and cortical neurons can be achieved by single spike cellular electrophysiology using electrodes alongside GEI imaging (Xiao et al. 2017). MBE lacks the ability to currently import and manage spike data. Since a colleague has already designed a program for this purpose this functionality was not prioritized for inclusion in MBE (Swindale and Spacek 2014).

Directed Connectivity

A nonzero phase lag in connectivity does not necessarily imply a causal or directed relationship (M. X. Cohen 2014). As phase-lagged connectivity (high synchrony) may appear between regions A and B without a causal or even direct interaction between them if both A and B's activity are causally linked with region C (See Fig. 4.3). Moreover, given activity between regions A and B do have high synchrony, it may be difficult to assess which region's activity precedes the other (See Fig. 4.3). Directed functional connectivity analysis is required, of which, Granger Causality (GC) is the prime example (Granger 1969; M. X. Cohen 2014; Goebel et al. 2003; Harrison, Penny, and K. Friston 2003; Karl Friston, R. Moran, and Anil K Seth 2013). GC can dissociate an A to B connection from a B to A one. GC can also be multivariate, although the simpler bivariate form is more commonly used (M. X. Cohen 2014; Karl Friston, R. Moran, and Anil K Seth 2013). GC is however computationally time-consuming to perform and doubles the number of

statistical comparisons that need to be controlled for and thus might be tedious for large-scale exploratory analysis (M. X. Cohen 2014; Rajapakse and Zhou 2007; Karl Friston, R. Moran, and Anil K Seth 2013). Finally, GC is model-based and thus requires an *a priori model* to begin with (Rajapakse and Zhou 2007; Karl Friston, R. Moran, and Anil K Seth 2013).

Bayesian Network (BN)s can be used to model hidden sources of the hypothesis space for directional connectivity and are thus a data-driven alternative to GC (Karl Friston, R. Moran, and Anil K Seth 2013). BNs can be used to learn large or unexplored cognitive networks from brain imaging data by assuming that the basis of such networks does not have a proper prior model. (Zheng and Rajapakse 2006). GC, by contrast, is used when a previously known or hypothesized neural system model is valid rather than to 'find' a suitable model from the data (Suhr 2006). BNs however do not provide an explicit mechanism to represent temporal dependencies among multiple processes at brain regions and instead give one snapshot of brain connectivity, taking into consideration the whole experiment (Rajapakse and Zhou 2007). Therefore, BN may often report directionality that is indeterminate or bi-directional (Chickering 1995). This in turn limits the causal inference one can derive from BNs (Rajapakse and Zhou 2007) and they remain best suited to functional connectivity analysis rather than effective connectivity analysis (see Sect. 4.2.2).

The implementation of GC or BNs in MBE would require a visualization of multivariate network graphs. The plugin for such a visualization would likely include UI elements to allow for the measurement of node centrality via different algorithms such that there are measures of the most important vertices within the graph (Silva et al. 2017). Graphs are generally useful for providing summary information regarding large-scale or multivariate network dynamics (M. X. Cohen 2014). Work has previously been published by our lab where graphs have been used to display data from correlation matrices in a different form (Connor et al. 2016; Lim, J. LeDue, et al. 2013) or results from large-scale cortical mapping using CHR2 stimulation to establish effective connectivity (Lim, Majid H. Mohajerani, et al. 2012). It therefore makes the most sense for graph visualization to first be implemented and validated on correlation matrices before implementing GC or BNs. However, the best framework for a graph-based visualization in MBE is not clear. The pyqtgraph framework (Campagnola 2016) MBE is built on top of does provide a framework wherein nodes, vertices etc. can be described and made interactive within a scene. However, the framework does not specify the location of nodes. This needs to be specified in advance. No clear mechanism exists either via this framework for moving individual nodes and keeping the graph intact in case a user wants to re-arrange the location of nodes in a graph to suit their analysis. Various other Python graph visualization frameworks exist that might be better suited such as NodeBox, NetworkX, matplotlib or Graphviz (De Bleser and De Smedt n.d.; Hunter 2007; Hagberg, Swart, and S Chult 2008; Gansner and North 2000). Which framework to adopt remains an open question and thus MBE remains constrained to undirected connectivity. Directed connectivity is only possible in MBE when sensory-evoked data is used. Therefore, if claims about directionality or causality are important sensory evoked mapping, such as CHR2-evoked mapping, is preferred. As both directionality and causality are important for fully establishing the functional mesoscale connectome, spontaneous activity mapping cannot presently entirely replace sensory-evoked mapping.

Functional vs Effective Connectivity

The distinction between functional and effective connectivity is analogous to the distinction between correlation and causation. Functional connectivity refers to linear or nonlinear co-variation between fluctuations in activity recorded from distinct neural networks. Effective connectivity refers to a causal influence of activity in one neural network over activity in another neural network (Karl J. Friston 1994; M. X. Cohen 2014). Effective connectivity is always directed (Karl Friston, R. Moran, and Anil K Seth 2013). Although effective connectivity is typically established via controlled sensory-evoked experiments, this need not be the case. Effective connectivity can rest on an explicit (parameterised) model of causal influences — usually expressed in terms of difference (discrete time) or differential (continuous time) equations (Karl Friston, R. Moran, and Anil K Seth 2013). The most popular methods implementing such equations are Dynamic Causal Modelling (DCM) and Structural Equation Modelling (SEM), with DCM overtaking SEM in recent years (Karl Friston, R. Moran, and Anil K Seth 2013; Zheng and Rajapakse 2006; McIntosh et al. 1994; Nyberg et al. 1996; Bavelier et al. 2000; Honey et al. 2002; Nezafat, Shadmehr, and Holcomb 2001; Petersson et al. 2000; Büchel and K. J. Friston 1997; David et al. 2006; Garrido, Kilner, Kiebel, and Karl J. Friston 2007; Kiebel, Garrido, and Karl J. Friston 2007; Garrido, Kilner, Kiebel, Stephan, et al. 2009; R. J. Moran et al. 2011; Boly et al. 2011). DCM and SEM are both confirmatory (i.e model-based) methods and are thus only useful when a prior connectivity model is available (Rajapakse and Zhou 2007). Note that GC is often cited as an effective connectivity method, however GC's model is about making inferences about statistical dependencies over time as modelled with an autoregressive process, not about the causal coupling (Karl J Friston 2011). As such it may be better regarded as lagged/directional functional activity (Karl J Friston 2011). Dynamic Bayesian Network (DBN)s are a data-driven alternative to DCM and SEM that can characterize the effective connectivity among brain regions in a complete statistical sense without an underlying hypothesis (Rajapakse and Zhou 2007). All these methods tend to struggle with inter-subject variability of brain connectivity in group studies without destroying inter-group differences (X. Chen, Z. J. Wang, and M. J. McKeown 2010; Rajapakse and Zhou 2007). This is problematic if a general connectome is to be established that isn't biased according to subjects used. Regression analysis techniques using Last Absolute Shrinkage and Selection Operator (LASSO) have been developed to address this concern (X. Chen, Z. J. Wang, and M. J. McKeown 2010). DCM, SEM, DBN and LASSO would all require directed graph visualizations if they are ever to be implemented in MBE. Thus MBE remains constrained to functional connectivity unless sensory-evoked experiments are employed. Furthermore, all these methods are firmly established in human FMRI studies on spontaneous brain data. Their applicability to being applied directly to mesoscale brain analysis of spontaneous mouse data as is done in our lab remains unclear.

Pearson Correlation Coefficient Limitations

Thus far we have covered five limitations - that MBE currently only deals with model-based bivariate unimodal undirected functional analysis. However, even with this specificity there are still many techniques in this category such as the use of different correlation coefficients including Spearman and Pearson, or cross-correlation or mutual information. MBE only makes use of Pearson correlation coefficients thus far, limiting it even further.

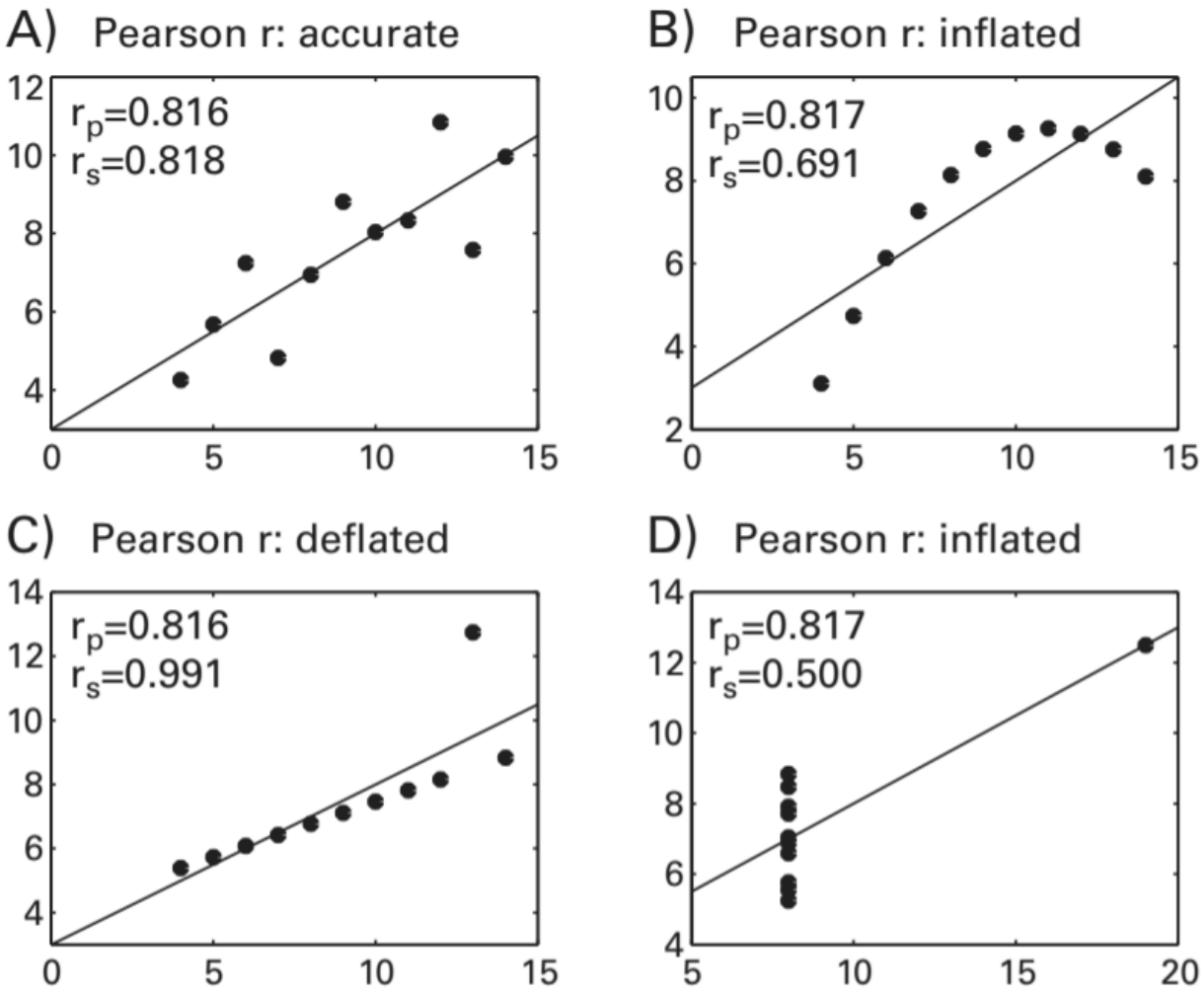


Figure 4.4: Comparison of Pearson (r_p) and Spearman (r_s) correlation coefficients for Anscombe's quartet (Chatterjee and Firat 2007). Data for A are normally distributed. Figure copied with permission from The MIT Press. Mike X. Cohen (2014). *Analyzing Neural Time Series Data: Theory and Practice*. English. 1 edition. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-01987-3.

Pearson correlation relies on the assumption that data are normally distributed. Violations of this assumption introduce bias (M. X. Cohen 2014). Spearman correlation is identical to Pearson, except that it involves rank-transforming the data before applying the Pearson equation. e.g. numbers 0.1, 0.2, 0.21, 10,000.1, become 1, 2, 3, 4. Normally, 10,000.1 in this context would be an outlier, but ranking the variables eliminates the influence of this outlier without removing it from the data (M. X. Cohen 2014). This makes it nonparametric. Pearson can be inflated or deflated depending on the leverage of outliers or if data are non-normally distributed, otherwise Spearman and Pearson are near-identical (Fig. 4.4) (Chatterjee and Firat 2007). Spearman thus does provide a less biased correlation and is especially useful for data where data is typically non-normal such as inter-subject correlation or across temporal frequency bands. Since

FLAs (primary users of MBE) plan to do analysis in both these areas the option to use Spearman in place of Pearson will soon be implemented for all analysis in MBE. Pearson is still included as an option as it is often preferred regardless because it emphasizes large events (e.g. large $\Delta F/F_0$ fluctuations) which contribute more to the r-value than the noise (weak $\Delta F/F_0$ fluctuations).

Cross-correlation is a measure of similarity of two series as a function of displacement of one relative to the other (*Cross-correlation* 2017). The similarity function used is typically a dot product, though it could be any bivariate measure such as Pearson or Spearman (better termed a lagged correlation). CCA already has applications to minimizing the limitations of bivariate analysis as previously discussed and already sees wide use in our lab (Vanni and Timothy H. Murphy 2014; Vanni, Chan, et al. 2017; Majid H. Mohajerani, Chan, et al. 2013; Xie et al. 2016; Xiao et al. 2017; Silasi, Xiao, et al. 2016). It has particular applications for the identification of cortical motifs through exploratory analysis where the user sets a similarity function and a threshold for this function and explores lags for where the threshold is reached across an image stack (Majid H. Mohajerani, Chan, et al. 2013; Chan et al. 2015).

MI is a simple but robust method for detecting shared information between two or more (multivariate extension) variables (M. X. Cohen 2014). MI, unlike Pearson correlation, can detect many kinds of relationships including nonlinear and linear ones. A circle for example has a Pearson correlation of zero, but a nonzero MI (M. X. Cohen 2014). That being said, this measure provides no information about whether the relationship is linear or nonlinear or even whether it is positive or negative - only a single value is reported. As has previously been discussed, MI is used by certain forms of ICA, a computational method that we are exploring for separating spontaneous brain activity signals into additive sub-components, and thus a configurable MI plugin will likely see eventual implementation in MBE.

Hemodynamic Correction

The contribution of hemodynamic cross-talk to a GEI or dye signal can be removed by three methods. In the most naive approach the properties of the tissue (and changes in absorption) are assume to be the same at emission and excitation wavelengths. Contamination removal (under these approximations) can then be done by simply dividing the fluorescence ratio by the reflectance ratio (Y. Ma et al. 2016). MBE supports this method but not the further more rigorous methods by the use of blind source separation techniques such as PCA or excitation and emission attenuation (Y. Ma et al. 2016).

4.3 Concluding Remarks

In its current incarnation MBE meets most recommended design features for improved reproducible software and therefore as a method agnostic tool has strong suitability to be used to process neurophotonic optical imaging data including, but not limited to IOS, LSI, VSD and most GEIs. Although it is limited in only supporting model-based bivariate unimodal undirected functional connectivity analyses via Pearson correlation coefficients.

Given many of MBE's limitations can be easily implemented with the extensible framework it offers MBE's strengths outweigh its weaknesses, in theory. The FLA who is the primary user of tool uses it daily, sometimes with difficulty and other times with no difficulty. The overwhelming source of difficulties are due

to errors introduced after a modification or extension that could easily be avoided if a continuous integration testing framework had been used. We acknowledge this as MBE's single largest weakness as it directly weakens the case for both Aim 1 and Aim 2 being met and it must be addressed moving forward. Although no case study is provided validating improved research performance of MBE users Aim 1 is met as the gatekeeper (principal investigator) is satisfied. Improvements can doubtless be made iteratively to improve the user experience. The multiple pipelines automation feature for instance should be re-designed so that a tabbed interface exists for multiple pipelines rather than having all pipelines in a single list in the UI. MBE's internal design can likewise be further iteratively improved, but as a framework that offers undergraduate programmers research-grade software that abides by basic software engineering principles (encapsulation, inheritance, polymorphism (3.1.1)) Aim 2 is also considered met.

Bibliography

Abdelfattah, Ahmed S. et al. (2016). "A Bright and Fast Red Fluorescent Protein Voltage Indicator That Reports Neuronal Activity in Organotypic Brain Slices". en. In: *Journal of Neuroscience* 36.8, pp. 2458–2472. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3484-15.2016. URL: http://www.jneurosci.org/content/36/8/2458 (visited on 05/26/2017).

Ackman, James B., Timothy J. Burbridge, and Michael C. Crair (2012). "Retinal waves coordinate patterned activity throughout the developing visual system". eng. In: *Nature* 490.7419, pp. 219–225. ISSN: 1476-4687. DOI: 10.1038/nature11529.

Ahrens, Misha B., Jennifer M. Li, et al. (2012). "Brain-wide neuronal dynamics during motor adaptation in zebrafish". en. In: *Nature* 485.7399, pp. 471–477. ISSN: 0028-0836. DOI: 10.1038/nature11057. URL: https://www.nature.com/nature/journal/v485/n7399/full/nature11057.html (visited on 05/26/2017).

Ahrens, Misha B., Michael B. Orger, et al. (2013). "Whole-brain functional imaging at cellular resolution using light-sheet microscopy". en. In: *Nature Methods* 10.5, pp. 413–420. ISSN: 1548-7091. DOI: 10.1038/nmeth.2434. URL: http://www.nature.com/nmeth/journal/v10/n5/full/nmeth.2434.html (visited on 05/26/2017).

Ajetunmobi, A. et al. (2014). "Nanotechnologies for the study of the central nervous system". In: *Progress in Neurobiology* 123, pp. 18–36. ISSN: 0301-0082. DOI: 10.1016/j.pneurobio.2014.09.004. URL: http://www.sciencedirect.com/science/article/pii/S030100821400104X (visited on 11/27/2016).

Akemann, Walther et al. (2013). "Two-photon voltage imaging using a genetically encoded voltage indicator". en. In: *Scientific Reports* 3, p. 2231. ISSN: 2045-2322. DOI: 10.1038/srep02231. URL: http://www.nature.com/srep/2013/130719/srep02231/full/srep02231.html (visited on 05/26/2017).

Akerboom, Jasper, Nicole Carreras Calderón, et al. (2013). "Genetically encoded calcium indicators for multi-color neural activity imaging and combination with optogenetics". eng. In: *Frontiers in Molecular Neuroscience* 6, p. 2. DOI: 10.3389/fnmol.2013.00002.

Akerboom, Jasper, Jonathan D. Vélez Rivera, et al. (2009). "Crystal structures of the GCaMP calcium sensor reveal the mechanism of fluorescence signal change and aid rational design". eng. In: *The Journal of Biological Chemistry* 284.10, pp. 6455–6464. ISSN: 0021-9258. DOI: 10.1074/jbc.M807657200.

Aldama, Eduardo Ferro. *Python SOLID*. Technology. URL: https://www.slideshare.net/DrTrucho/python-solid.

Allen Mouse Brain Connectivity Atlas (2012). http://connectivity.brain-map.org/. URL: http://connectivity.brain-map.org/.

Alsheikh-Ali, Alawi A. et al. (2011). “Public Availability of Published Research Data in High-Impact Journals”. In: *PLOS ONE* 6.9, e24357. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0024357. URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0024357 (visited on 05/26/2017).

Anders, K. and Herbert A. Simon (1980). “Verbal reports as data”. English. In: *Psychological Review* 87.3, pp. 215–251. ISSN: 1939-1471 0033-295X. DOI: 10.1037/0033-295X.87.3.215.

Arenkiel, Benjamin R. et al. (2007). “In Vivo Light-Induced Activation of Neural Circuitry in Transgenic Mice Expressing Channelrhodopsin-2”. In: *Neuron* 54.2, pp. 205–218. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2007.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0896627307001833 (visited on 08/30/2016).

Asanuma, H., A. Arnold, and P. Zarzecki (1976). “Further study on the excitation of pyramidal tract cells by intracortical microstimulation”. eng. In: *Experimental Brain Research* 26.5, pp. 443–461. ISSN: 0014-4819.

Axelrod, Vadim (2014). “Minimizing bugs in cognitive neuroscience programming”. English. In: *Frontiers in Psychology* 5. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2014.01435. URL: http://journal.frontiersin.org/article/10.3389/fpsyg.2014.01435/full (visited on 07/01/2017).

Axmacher, Nikolai et al. (2010). “Cross-frequency coupling supports multi-item working memory in the human hippocampus”. en. In: *Proceedings of the National Academy of Sciences* 107.7, pp. 3228–3233. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0911531107. URL: http://www.pnas.org/content/107/7/3228 (visited on 07/13/2017).

Basak, Kausik, M. Manjunatha, and Pranab Kumar Dutta (2012). “Review of laser speckle-based analysis in medical imaging”. eng. In: *Medical & Biological Engineering & Computing* 50.6, pp. 547–558. ISSN: 1741-0444. DOI: 10.1007/s11517-012-0902-z.

Baumgartner, R. et al. (2000). “Comparison of two exploratory data analysis methods for fMRI: fuzzy clustering vs. principal component analysis”. eng. In: *Magnetic Resonance Imaging* 18.1, pp. 89–94. ISSN: 0730-725X.

Bavelier, D. et al. (2000). “Visual attention to the periphery is enhanced in congenitally deaf individuals”. eng. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 20.17, RC93. ISSN: 0270-6474.

Beckmann, Christian F. and Stephen M. Smith (2004). “Probabilistic independent component analysis for functional magnetic resonance imaging”. eng. In: *IEEE transactions on medical imaging* 23.2, pp. 137–152. ISSN: 0278-0062. DOI: 10.1109/TMI.2003.822821.

Begley, C. Glenn and Lee M. Ellis (2012). “Drug development: Raise standards for preclinical cancer research”. eng. In: *Nature* 483.7391, pp. 531–533. ISSN: 1476-4687. DOI: 10.1038/483531a.

Berger, Thomas et al. (2007). “Combined voltage and calcium epifluorescence imaging in vitro and in vivo reveals subthreshold and suprathreshold dynamics of mouse barrel cortex”. eng. In: *Journal of Neurophysiology* 97.5, pp. 3751–3762. ISSN: 0022-3077. DOI: 10.1152/jn.01178.2006.

Berntsson, Ronnie P.-A. et al. (2010). “A structural classification of substrate-binding proteins”. eng. In: *FEBS letters* 584.12, pp. 2606–2617. ISSN: 1873-3468. DOI: 10.1016/j.febslet.2010.04.043.

Betley, J. Nicholas et al. (2015). “Neurons for hunger and thirst transmit a negative-valence teaching signal”. en. In: *Nature* 521.7551, pp. 180–185. ISSN: 0028-0836. DOI: 10.1038/nature14416. URL: https://www.nature.com/nature/journal/v521/n7551/full/nature14416.html (visited on 05/26/2017).

Biessmann, Felix et al. (2011). “Analysis of multimodal neuroimaging data”. eng. In: *IEEE reviews in biomedical engineering* 4, pp. 26–58. ISSN: 1941-1189. DOI: 10.1109/RBME.2011.2170675.

Blasdel, Gary G. and Guy Salama (1986). “Voltage-sensitive dyes reveal a modular organization in monkey striate cortex”. en. In: *Nature* 321.6070, pp. 579–585. ISSN: 0028-0836. DOI: 10.1038/321579a0. URL: https://www.nature.com/nature/journal/v321/n6070/abs/321579a0.html (visited on 05/26/2017).

Boas, David A. and Andrew K. Dunn (2010). “Laser speckle contrast imaging in biomedical optics”. In: *Journal of Biomedical Optics* 15.1. ISSN: 1083-3668. DOI: 10.1117/1.3285504. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2816990/ (visited on 05/11/2017).

Bodnar, Jan (2015). *PyQt4 tutorial - learn GUI programming with PyQt4*. English. URL: http://zetcode.com/gui/pyqt4/ (visited on 07/14/2017).

Boehm, B. W. (1987). “Software Process Management: Lessons Learned from History”. In: *Proceedings of the 9th International Conference on Software Engineering*. ICSE ’87. Los Alamitos, CA, USA: IEEE Computer Society Press, pp. 296–298. ISBN: 978-0-89791-216-7. URL: http://dl.acm.org/citation.cfm?id=41765.41798 (visited on 05/26/2017).

— (1988). “A spiral model of software development and enhancement”. In: *Computer* 21.5, pp. 61–72. ISSN: 0018-9162. DOI: 10.1109/2.59.

Bohland, J. W. et al. (2009). “A proposal for a coordinated effort for the determination of brainwide neuroanatomical connectivity in model organisms at a mesoscopic scale”. eng. In: *PLoS Comput Biol* 5, e1000334. ISSN: 1553-7358 (Electronic) 1553-734X (Linking). DOI: 10.1371/journal.pcbi.1000334.

Boly, Melanie et al. (2011). “Preserved Feedforward But Impaired Top-Down Processes in the Vegetative State”. en. In: *Science* 332.6031, pp. 858–862. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1202043. URL: http://science.sciencemag.org/content/332/6031/858 (visited on 07/05/2017).

Borkin, Michelle A. et al. (2011). “Evaluation of Artery Visualizations for Heart Disease Diagnosis”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.

Bota, Mihail, Olaf Sporns, and Larry W. Swanson (2015). “Architecture of the cerebral cortical association connectome underlying cognition”. eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 112.16, E2093–2101. ISSN: 1091-6490. DOI: 10.1073/pnas.1504394112.

Bota, Mihail and Larry W. Swanson (2007). “The neuron classification problem”. eng. In: *Brain Research Reviews* 56.1, pp. 79–88. ISSN: 0165-0173. DOI: 10.1016/j.brainresrev.2007.05.005.

Bouchard, Matthew B. et al. (2009). “Ultra-fast multispectral optical imaging of cortical oxygenation, blood flow, and intracellular calcium dynamics”. eng. In: *Optics Express* 17.18, pp. 15670–15678. ISSN: 1094-4087.

Boyden, Edward S. et al. (2005). “Millisecond-timescale, genetically targeted optical control of neural activity”. en. In: *Nature Neuroscience* 8.9, pp. 1263–1268. ISSN: 1097-6256. DOI: 10.1038/nn1525. URL: http://www.nature.com/neuro/journal/v8/n9/full/nn1525.html (visited on 05/25/2017).

Brown, Barry, Stuart Reeves, and Scott Sherwood (2011). “Into the Wild: Challenges and Opportunities for Field Trial Methods”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. New York, NY, USA: ACM, pp. 1657–1666. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979185. URL: http://doi.acm.org/10.1145/1978942.1979185.

Brown, Ramsay A. and Larry W. Swanson (2013). “Neural systems language: a formal modeling language for the systematic description, unambiguous communication, and automated digital curation of neural connectivity”. eng. In: *The Journal of Comparative Neurology* 521.13, pp. 2889–2906. ISSN: 1096-9861. DOI: 10.1002/cne.23348.

Büchel, C. and K. J. Friston (1997). “Modulation of connectivity in visual pathways by attention: cortical interactions evaluated with structural equation modelling and fMRI”. eng. In: *Cerebral Cortex (New York, N.Y.: 1991)* 7.8, pp. 768–778. ISSN: 1047-3211.

Bumstead, Jonathan R. et al. (2016). “Cerebral functional connectivity and Mayer waves in mice: Phenomena and separability”. eng. In: *Journal of Cerebral Blood Flow and Metabolism: Official Journal of the International Society of Cerebral Blood Flow and Metabolism*. ISSN: 1559-7016. DOI: 10.1177/0271678X16629977.

Button, Katherine S. et al. (2013). “Power failure: why small sample size undermines the reliability of neuroscience”. en. In: *Nature Reviews Neuroscience* 14.5, pp. 365–376. ISSN: 1471-003X. DOI: 10.1038/nrn3475. URL: http://www.nature.com/nrn/journal/v14/n5/full/nrn3475.html (visited on 05/04/2017).

Calhoun, V. D. et al. (2001). “Spatial and temporal independent component analysis of functional MRI data containing a pair of task-related waveforms”. eng. In: *Human Brain Mapping* 13.1, pp. 43–53. ISSN: 1065-9471.

Calhoun, Vince D., Jingyu Liu, and Tülay Adali (2009). “A review of group ICA for fMRI data and ICA for joint inference of imaging, genetic, and ERP data”. eng. In: *NeuroImage* 45.1 Suppl, S163–172. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2008.10.057.

Campagnola, Luke (2016). *PyQtGraph - Scientific Graphics and GUI Library for Python*. http://www.pyqtgraph.org/. URL: http://www.pyqtgraph.org/ (visited on 09/09/2016).

Cao, Guan et al. (2013). “Genetically Targeted Optical Electrophysiology in Intact Neural Circuits”. English. In: *Cell* 154.4, pp. 904–913. ISSN: 0092-8674, 1097-4172. DOI: 10.1016/j.cell.2013.07.027. URL: http://www.cell.com/cell/abstract/S0092-8674(13)00898-2 (visited on 05/26/2017).

Carandini, Matteo et al. (2015). “Imaging the Awake Visual Cortex with a Genetically Encoded Voltage Indicator”. en. In: *The Journal of Neuroscience* 35.1, pp. 53–63. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0594-14.2015. URL: http://www.jneurosci.org/content/35/1/53 (visited on 08/29/2016).

Chan, Allen W. et al. (2015). “Mesoscale infraslow spontaneous membrane potential fluctuations recapitulate high-frequency activity cortical motifs”. eng. In: *Nature Communications* 6, p. 7738. ISSN: 2041-1723. DOI: 10.1038/ncomms8738.

Chatterjee, Sangit and Aykut Firat (2007). “Generating Data with Identical Statistics but Dissimilar Graphics”. In: *The American Statistician* 61.3, pp. 248–254. ISSN: 0003-1305. DOI: 10.1198/000313007X220057. URL: http://dx.doi.org/10.1198/000313007X220057.

Chen, Tsai-Wen et al. (2013). “Ultrasensitive fluorescent proteins for imaging neuronal activity”. en. In: *Nature* 499.7458, pp. 295–300. ISSN: 0028-0836. DOI: 10.1038/nature12354. URL: http://www.nature.com/nature/journal/v499/n7458/full/nature12354.html (visited on 12/08/2015).

Chen, X., Z. J. Wang, and M. McKeown (2016). “Joint Blind Source Separation for Neurophysiological Data Analysis: Multiset and multimodal methods”. In: *IEEE Signal Processing Magazine* 33.3, pp. 86–107. ISSN: 1053-5888. DOI: 10.1109/MSP.2016.2521870.

Chen, X., Z. J. Wang, and M. J. McKeown (2010). “FMRI group studies of brain connectivity via a group robust Lasso”. In: *2010 IEEE International Conference on Image Processing*, pp. 589–592. DOI: 10.1109/ICIP.2010.5652779.

Chiang, Ann-Shyn et al. (2011). “Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution”. eng. In: *Current biology: CB* 21.1, pp. 1–11. ISSN: 1879-0445. DOI: 10.1016/j.cub.2010.11.056.

Chickering, David Maxwell (1995). “A Transformational Characterization of Equivalent Bayesian Network Structures”. In: *Microsoft Research*. URL: https://www.microsoft.com/en-us/research/publication/a-transformational-characterization-of-equivalent-bayesian-network-structures/ (visited on 07/05/2017).

Cohen, Mike X. (2014). *Analyzing Neural Time Series Data: Theory and Practice*. English. 1 edition. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-262-01987-3.

Collette, Andrew et al. (2017). *HDF5 for Python – The h5py package is a Pythonic interface to the HDF5 binary data format*. original-date: 2012-09-21T00:40:02Z. URL: https://github.com/h5py/h5py.

Connor, Steven A. et al. (2016). “Altered Cortical Dynamics and Cognitive Function upon Haploinsufficiency of the Autism-Linked Excitatory Synaptic Suppressor MDGA2”. eng. In: *Neuron* 91.5, pp. 1052–1068. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2016.08.016.

Correa, Nicolle M. et al. (2010). “Canonical Correlation Analysis for Data Fusion and Group Inferences”. In: *IEEE signal processing magazine* 27.4, pp. 39–50. ISSN: 1053-5888. DOI: 10.1109/MSP.2010.936725. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2919827/.

Coverity scan open source report 2014 (2015). Tech. rep. Synopsys. URL: http://go.coverity.com/rs/157-LQW-289/images/2014-Coverity-Scan-Report.pdf (visited on 07/01/2017).

Crochet, Sylvain and Carl C. H. Petersen (2006). “Correlating whisker behavior with membrane potential in barrel cortex of awake mice”. en. In: *Nature Neuroscience* 9.5, pp. 608–610. ISSN: 1097-6256. DOI: 10.1038/nn1690. URL: http://www.nature.com/neuro/journal/v9/n5/full/nn1690.html (visited on 08/30/2016).

Crocker, Jennifer and M. Lynne Cooper (2011). “Addressing Scientific Fraud”. en. In: *Science* 334.6060, pp. 1182–1182. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1216775. URL: http://science.sciencemag.org/content/334/6060/1182 (visited on 06/28/2017).

Cross-correlation (2017). en. Page Version ID: 787125942. URL: https://en.wikipedia.org/w/index.php?title=Cross-correlation&oldid=787125942.

Dana, Hod et al. (2016). "Sensitive red protein calcium indicators for imaging neural activity". en. In: *eLife* 5, e12727. ISSN: 2050-084X. DOI: 10.7554/eLife.12727. URL: https://elifesciences.org/content/5/e12727v2 (visited on 05/26/2017).

Dance, Amber (2015). "Neuroscience: Connectomes make the map". en. In: *Nature* 526.7571, pp. 147–149. ISSN: 0028-0836. DOI: 10.1038/526147a. URL: https://www.nature.com/nature/journal/v526/n7571/full/526147a.html (visited on 04/28/2017).

Daniel, Andy G. S. et al. (2015). "Optical electrocorticogram (OECoG) using wide-field calcium imaging reveals the divergence of neuronal and glial activity during acute rodent seizures". eng. In: *Epilepsy & Behavior: E&B* 49, pp. 61–65. ISSN: 1525-5069. DOI: 10.1016/j.yebeh.2015.04.036.

David, Olivier et al. (2006). "Dynamic causal modeling of evoked responses in EEG and MEG". eng. In: *NeuroImage* 30.4, pp. 1255–1272. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2005.10.045.

De Bleser, Frederik and Tom De Smedt. "NodeBox". In: URL: http://nodebox.net/code/index.php/Home.

Deisseroth, Karl (2011). "Optogenetics". en. In: *Nature Methods* 8.1, pp. 26–29. ISSN: 1548-7091. DOI: 10.1038/nmeth.f.324. URL: https://www.nature.com/nmeth/journal/v8/n1/full/nmeth.f.324.html (visited on 07/11/2017).

Devor, A. et al. (2013). "The challenge of connecting the dots in the B.R.A.I.N". eng. In: *Neuron* 80, pp. 270–4. ISSN: 1097-4199 (Electronic) 0896-6273 (Linking). DOI: 10.1016/j.neuron.2013.09.008S0896-6273(13)00806-4[pii].

Ding, JingJin et al. (2014). "Structural basis of the ultrasensitive calcium indicator GCaMP6". eng. In: *Science China. Life Sciences* 57.3, pp. 269–274. ISSN: 1869-1889. DOI: 10.1007/s11427-013-4599-5.

Dombeck, Daniel A. et al. (2010). "Functional imaging of hippocampal place cells at cellular resolution during virtual navigation". en. In: *Nature Neuroscience* 13.11, pp. 1433–1440. ISSN: 1097-6256. DOI: 10.1038/nn.2648. URL: http://www.nature.com/neuro/journal/v13/n11/full/nn.2648.html (visited on 05/26/2017).

Dublon, I. A. N. et al. (2016). "Scintillate: An open-source graphical viewer for time-series calcium imaging evaluation and pre-processing". In: *Journal of Neuroscience Methods* 273, pp. 120–127. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2016.08.010. URL: http://www.sciencedirect.com/science/article/pii/S0165027016301881 (visited on 12/23/2016).

Dunn, Timothy W. et al. (2016). "Brain-wide mapping of neural activity controlling zebrafish exploratory locomotion". en. In: *eLife* 5, e12741. ISSN: 2050-084X. DOI: 10.7554/eLife.12741. URL: https://elifesciences.org/content/5/e12741v2 (visited on 05/26/2017).

Eddins, Steve (2017). *A New Colormap for MATLAB – Part 1 – Introduction » Steve on Image Processing*. https://blogs.mathworks.com/steve/2014/10/13/a-new-colormap-for-matlab-part-1-introduction/. URL: https://blogs.mathworks.com/steve/2014/10/13/a-new-colormap-for-matlab-part-1-introduction/ (visited on 01/09/2017).

Eglen, Stephen J. et al. (2017). "Toward standard practices for sharing computer code and programs in neuroscience". en. In: *Nature Neuroscience* 20.6, pp. 770–773. ISSN: 1097-6256. DOI:

10.1038/nn.4550. URL: http://www.nature.com/neuro/journal/v20/n6/full/nn.4550.html (visited on 05/26/2017).

Esposito, Fabrizio et al. (2002). "Spatial independent component analysis of functional MRI time-series: to what extent do results depend on the algorithm used?" eng. In: *Human Brain Mapping* 16.3, pp. 146–157. ISSN: 1065-9471. DOI: 10.1002/hbm.10034.

Ferezou, I. et al. (2007). "Spatiotemporal dynamics of cortical sensorimotor integration in behaving mice". eng. In: *Neuron* 56, pp. 907–23. ISSN: 0896-6273 (Print) 0896-6273 (Linking). DOI: S0896-6273(07)00763-5[pii]10.1016/j.neuron.2007.10.007.

Flotho, P. et al. (2016). "Motion invariant contrast enhancement of optical imaging data in the gradient domain". In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 3937–3940. DOI: 10.1109/EMBC.2016.7591588.

Fosque, Benjamin F. et al. (2015). "Neural circuits. Labeling of active neural circuits in vivo with designed calcium integrators". eng. In: *Science (New York, N.Y.)* 347.6223, pp. 755–760. ISSN: 1095-9203. DOI: 10.1126/science.1260922.

Fox, Michael D. and Michael Greicius (2010). "Clinical Applications of Resting State Functional Connectivity". In: *Frontiers in Systems Neuroscience* 4. ISSN: 1662-5137. DOI: 10.3389/fnsys.2010.00019. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2893721/ (visited on 11/27/2016).

Fox, Michael D., Abraham Z. Snyder, et al. (2005). "The human brain is intrinsically organized into dynamic, anticorrelated functional networks". en. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.27, pp. 9673–9678. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0504136102. URL: http://www.pnas.org/content/102/27/9673 (visited on 01/10/2017).

Fox, Michael D., Dongyang Zhang, et al. (2009). "The global signal and observed anticorrelated resting state brain networks". eng. In: *Journal of Neurophysiology* 101.6, pp. 3270–3283. ISSN: 0022-3077. DOI: 10.1152/jn.90777.2008.

Friston, Karl J. (1994). "Functional and effective connectivity in neuroimaging: A synthesis". en. In: *Human Brain Mapping* 2.1-2, pp. 56–78. ISSN: 1097-0193. DOI: 10.1002/hbm.460020107. URL: http://onlinelibrary.wiley.com/doi/10.1002/hbm.460020107/abstract (visited on 04/03/2017).

Friston, Karl J (2011). "Functional and effective connectivity: a review". In: *Brain connectivity* 1.1, pp. 13–36.

Friston, Karl, Rosalyn Moran, and Anil K Seth (2013). "Analysing connectivity with Granger causality and dynamic causal modelling". In: *Current Opinion in Neurobiology*. Macrocircuits 23.2, pp. 172–178. ISSN: 0959-4388. DOI: 10.1016/j.conb.2012.11.010. URL: http://www.sciencedirect.com/science/article/pii/S0959438812001845.

Fritsch, G and Eduard Hitzig (1870). *Ueber die elektrische Erregbarkeit des Grosshirns*. German. OCLC: 9353751. Leipzig: Veit.

Frostig, R. D. et al. (1990). "Cortical functional architecture and local coupling between neuronal activity and the microcirculation revealed by in vivo high-resolution optical imaging of intrinsic signals". eng.

In: *Proceedings of the National Academy of Sciences of the United States of America* 87.16, pp. 6082–6086. ISSN: 0027-8424.

Frostig, Ron D. and Cynthia H. Chen-Bee (2009). "Visualizing Adult Cortical Plasticity Using Intrinsic Signal Optical Imaging". eng. In: *In Vivo Optical Imaging of Brain Function*. Ed. by Ron D. Frostig. 2nd. Frontiers in Neuroscience. Boca Raton (FL): CRC Press/Taylor & Francis. ISBN: 978-1-4200-7684-4. URL: http://www.ncbi.nlm.nih.gov/books/NBK20227/ (visited on 01/09/2017).

Gamma, Erich et al. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. English. 1 edition. Reading, Mass: Addison-Wesley Professional. ISBN: 978-0-201-63361-0.

Gansner, Emden R. and Stephen C. North (2000). "An open graph visualization system and its applications to software engineering". In: *SOFTWARE - PRACTICE AND EXPERIENCE* 30.11, pp. 1203–1233.

Garnier, Simon, Noam Ross (Continuous scale), and Bob Rudis (Combined scales) (2016). *viridis: Default Color Maps from 'matplotlib'*. URL: https://cran.r-project.org/web/packages/viridis/index.html (visited on 01/09/2017).

Garrido, Marta I., James M. Kilner, Stefan J. Kiebel, and Karl J. Friston (2007). "Evoked brain responses are generated by feedback loops". en. In: *Proceedings of the National Academy of Sciences* 104.52, pp. 20961–20966. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0706274105. URL: http://www.pnas.org/content/104/52/20961 (visited on 07/05/2017).

Garrido, Marta I., James M. Kilner, Stefan J. Kiebel, Klaas E. Stephan, et al. (2009). "Repetition suppression and plasticity in the human brain". eng. In: *NeuroImage* 48.1, pp. 269–279. ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2009.06.034.

Ghosh, Kunal K. et al. (2011). "Miniaturized integration of a fluorescence microscope". en. In: *Nature Methods* 8.10, pp. 871–878. ISSN: 1548-7091. DOI: 10.1038/nmeth.1694. URL: http://www.nature.com/nmeth/journal/v8/n10/full/nmeth.1694.html (visited on 05/26/2017).

Gilb, Tom (1988). *Principles of Software Engineering Management*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 978-0-201-19246-9.

Glass, Robert L. and Tom DeMarco (2006). *Software Creativity 2.0*. English. Atlanta, Ga: Developer.* Books. ISBN: 978-0-9772133-1-3.

Goebel, Rainer et al. (2003). "Investigating directed cortical interactions in time-resolved fMRI data using vector autoregressive modeling and Granger causality mapping". In: *Magnetic Resonance Imaging* 21.10, pp. 1251–1261. ISSN: 0730-725X. DOI: 10.1016/j.mri.2003.08.026. URL: http://www.sciencedirect.com/science/article/pii/S0730725X03003370.

Gohlke, Christoph (2014). *tifffile - Read and write image data from and to TIFF files.* https://github.com/blink1073/tifffile. URL: https://github.com/blink1073/tifffile (visited on 11/19/2016).

Golay, X. et al. (1998). "A new correlation-based fuzzy logic clustering algorithm for fMRI". eng. In: *Magnetic Resonance in Medicine* 40.2, pp. 249–260. ISSN: 0740-3194.

Goodman, Steven N., Daniele Fanelli, and John P. A. Ioannidis (2016). "What does research reproducibility mean?" eng. In: *Science Translational Medicine* 8.341, 341ps12. ISSN: 1946-6242. DOI: 10.1126/scitranslmed.aaf5027.

Gore, Bryan B., Marta E. Soden, and Larry S. Zweifel (2014). “Visualization of plasticity in fear-evoked calcium signals in midbrain dopamine neurons”. eng. In: *Learning & Memory (Cold Spring Harbor, N.Y.)* 21.11, pp. 575–579. ISSN: 1549-5485. DOI: 10.1101/lm.036079.114.

Gorgolewski, Krzysztof J. and Russell A. Poldrack (2016). “A Practical Guide for Improving Transparency and Reproducibility in Neuroimaging Research”. eng. In: *PLoS biology* 14.7, e1002506. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1002506.

Granger, C. W. J. (1969). “Investigating Causal Relations by Econometric Models and Cross-spectral Methods”. In: *Econometrica* 37.3, pp. 424–438. ISSN: 0012-9682. DOI: 10.2307/1912791. URL: http://www.jstor.org/stable/1912791.

Greicius, Michael D. et al. (2004). “Default-mode network activity distinguishes Alzheimer’s disease from healthy aging: Evidence from functional MRI”. en. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.13, pp. 4637–4642. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0308627101. URL: http://www.pnas.org/content/101/13/4637 (visited on 07/03/2017).

Grinvald, A., R. D. Frostig, et al. (1988). “Optical imaging of neuronal activity”. eng. In: *Physiological Reviews* 68.4, pp. 1285–1366. ISSN: 0031-9333.

Grinvald, A., E. Lieke, et al. (1986). “Functional architecture of cortex revealed by optical imaging of intrinsic signals”. eng. In: *Nature* 324.6095, pp. 361–364. ISSN: 0028-0836. DOI: 10.1038/324361a0.

Grinvald, Amiram and Rina Hildesheim (2004). “VSDI: a new era in functional imaging of cortical dynamics”. en. In: *Nature Reviews Neuroscience* 5.11, pp. 874–885. ISSN: 1471-003X. DOI: 10.1038/nrn1536. URL: http://www.nature.com/nrn/journal/v5/n11/full/nrn1536.html (visited on 09/02/2016).

Grinvald, Amiram, Dahlia Sharon, et al. (2016). “Imaging the Neocortex Functional Architecture Using Multiple Intrinsic Signals: Implications for Hemodynamic-Based Functional Imaging”. eng. In: *Cold Spring Harbor Protocols* 2016.3, pdb.top089375. ISSN: 1559-6095. DOI: 10.1101/pdb.top089375.

Grynkiewicz, G., M. Poenie, and R. Y. Tsien (1985). “A new generation of Ca2+ indicators with greatly improved fluorescence properties”. eng. In: *The Journal of Biological Chemistry* 260.6, pp. 3440–3450. ISSN: 0021-9258.

Hagberg, Aric, Pieter Swart, and Daniel S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Laboratory (LANL).

Hagen, Espen et al. (2015). “ViSAPy: A Python tool for biophysics-based generation of virtual spiking activity for evaluation of spike-sorting algorithms”. In: *Journal of Neuroscience Methods* 245, pp. 182–204. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2015.01.029. URL: http://www.sciencedirect.com/science/article/pii/S0165027015000369 (visited on 11/27/2016).

Halchenko, Yaroslav O. and Michael Hanke (2015). “Four aspects to make science open “by design” and not as an after-thought”. In: *GigaScience* 4, p. 31. ISSN: 2047-217X. DOI: 10.1186/s13742-015-0072-7. URL: http://dx.doi.org/10.1186/s13742-015-0072-7.

Hamel, Elizabeth J. O. et al. (2015). “Cellular Level Brain Imaging in Behaving Mammals: An Engineering Approach”. In: *Neuron* 86.1, pp. 140–159. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2015.03.055. URL: http://www.sciencedirect.com/science/article/pii/S0896627315002792 (visited on 12/06/2015).

Haralick, Robert M. and Linda G. Shapiro (1992). *Computer and Robot Vision*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 978-0-201-56943-8.

Harris, J. A. et al. (2014). "Anatomical characterization of Cre driver mice for neural circuit mapping and manipulation". eng. In: *Front Neural Circuits* 8, p. 76. ISSN: 1662-5110 (Electronic) 1662-5110 (Linking). DOI: 10.3389/fncir.2014.00076.

Harrison, L., W. D. Penny, and K. Friston (2003). "Multivariate autoregressive modeling of fMRI time series". eng. In: *NeuroImage* 19.4, pp. 1477–1491. ISSN: 1053-8119.

Hartigan, John A. (1975). *Clustering Algorithms*. English. 1St Edition edition. New York: John Wiley & Sons Inc. ISBN: 978-0-471-35645-5.

Harvey, Christopher D., Philip Coen, and David W. Tank (2012). "Choice-specific sequences in parietal cortex during a virtual-navigation decision task". eng. In: *Nature* 484.7392, pp. 62–68. ISSN: 1476-4687. DOI: 10.1038/nature10918.

Harvey, Christopher D et al. (2009). "Intracellular dynamics of hippocampal place cells during virtual navigation". In: *Nature* 461.7266, pp. 941–946.

Haupt, Dirk (2017). *Mesoscale-Brain-Explorer: Repo for Mesoscale Brain Explorer (MBE)*. original-date: 2016-06-24T01:52:39Z. URL: https://github.com/Frikster/Mesoscale-Brain-Explorer.

Haupt, Dirk et al. (2017). "Mesoscale brain explorer, a flexible python-based image analysis and visualization tool". eng. In: *Neurophotonics* 4.3, p. 031210. ISSN: 2329-423X. DOI: 10.1117/1.NPh.4.3.031210.

Heim, Nicola et al. (2007). "Improved calcium imaging in transgenic mice expressing a troponin C-based biosensor". eng. In: *Nature Methods* 4.2, pp. 127–129. ISSN: 1548-7091. DOI: 10.1038/nmeth1009.

Helassa, Nordine et al. (2015). "Fast-Response Calmodulin-Based Fluorescent Indicators Reveal Rapid Intracellular Calcium Dynamics". en. In: *Scientific Reports* 5, p. 15978. ISSN: 2045-2322. DOI: 10.1038/srep15978. URL: http://www.nature.com/srep/2015/151103/srep15978/full/srep15978.html (visited on 05/26/2017).

Helmstaedter, Moritz (2013). "Cellular-resolution connectomics: challenges of dense neural circuit reconstruction". English. In: *Nature Methods; New York* 10.6, pp. 501–7. ISSN: 15487091. DOI: http://dx.doi.org.ezproxy.library.ubc.ca/10.1038/nmeth.2476. URL: http://search.proquest.com.ezproxy.library.ubc.ca/docview/1357394145/abstract/F7EF69C80EF84303PQ/1 (visited on 05/26/2017).

Herndon, Thomas, Michael Ash, and Robert Pollin (2014). "Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff". In: *Cambridge Journal of Economics* 38.2, pp. 257–279. ISSN: 0309-166X. DOI: 10.1093/cje/bet075. URL: https://academic.oup.com/cje/article-abstract/38/2/257/1714018/Does-high-public-debt-consistently-stifle-economic (visited on 07/01/2017).

Heroux, Michael A. and James M. Willenbring (2009). "Barely Sufficient Software Engineering: 10 Practices to Improve Your CSE Software". In: *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*. SECSE '09. Washington, DC, USA: IEEE Computer Society, pp. 15–21. ISBN: 978-1-4244-3737-5. DOI: 10.1109/SECSE.2009.5069157. URL: http://dx.doi.org/10.1109/SECSE.2009.5069157.

Herrick, C. Judson (1948). *The brain of the tiger salamander, Ambystoma tigrinum.* Chicago, Univ. of Chicago Press. URL: http://www.biodiversitylibrary.org/bibliography/6375.

Hikima, Takuya, Marianela Garcia-Munoz, and Gordon William Arbuthnott (2016). "Presynaptic D1 heteroreceptors and mGlu autoreceptors act at individual cortical release sites to modify glutamate release". eng. In: *Brain Research* 1639, pp. 74–87. ISSN: 1872-6240. DOI: 10.1016/j.brainres.2016.02.042.

Hildebrandt, Isabel J., Helen Su, and Wolfgang A. Weber (2008). "Anesthesia and Other Considerations for in Vivo Imaging of Small Animals". In: *ILAR Journal* 49.1, pp. 17–26. ISSN: 1084-2020. DOI: 10.1093/ilar.49.1.17. URL: https://academic.oup.com/ilarjournal/article/49/1/17/777307/Anesthesia-and-Other-Considerations-for-in-Vivo (visited on 05/26/2017).

Hillman, Elizabeth M. C. et al. (2007). "Depth-resolved optical imaging and microscopy of vascular compartment dynamics during somatosensory stimulation". eng. In: *NeuroImage* 35.1, pp. 89–104. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2006.11.032.

Hogg, M.C. (2013). *BMDanalyse - A graphical tool used for the regional analysis of a time series of 2D medical images*. https://github.com/mhogg/BMDanalyse. URL: https://github.com/mhogg/BMDanalyse (visited on 09/08/2016).

Honey, G. D. et al. (2002). "Effects of verbal working memory load on corticocortical connectivity modeled by path analysis of functional magnetic resonance imaging data". eng. In: *NeuroImage* 17.2, pp. 573–582. ISSN: 1053-8119.

Hosp, Jonas A. et al. (2008). "Thin-film epidural microelectrode arrays for somatosensory and motor cortex mapping in rat". eng. In: *Journal of Neuroscience Methods* 172.2, pp. 255–262. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2008.05.010.

Huber, Daniel et al. (2008). "Sparse optical microstimulation in barrel cortex drives learned behaviour in freely moving mice". en. In: *Nature* 451.7174, pp. 61–64. ISSN: 0028-0836. DOI: 10.1038/nature06445. URL: http://www.nature.com/nature/journal/v451/n7174/full/nature06445.html (visited on 08/30/2016).

Huber, D. et al. (2012). "Multiple dynamic representations in the motor cortex during sensorimotor learning". eng. In: *Nature* 484.7395, pp. 473–478. ISSN: 1476-4687. DOI: 10.1038/nature11039.

Hunnicutt, Barbara J. et al. (2014). "A comprehensive thalamocortical projection map at the mesoscopic level". ENG. In: *Nature Neuroscience* 17.9, pp. 1276–1285. ISSN: 1546-1726. DOI: 10.1038/nn.3780.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Hyvärinen, A. and E. Oja (2000). "Independent component analysis: algorithms and applications". eng. In: *Neural Networks: The Official Journal of the International Neural Network Society* 13.4-5, pp. 411–430. ISSN: 0893-6080.

Inc, Plotly Technologies (2015). *Collaborative data science*. URL: https://plot.ly.

Ioannidis, John P. A. et al. (2009). "Repeatability of published microarray gene expression analyses". eng. In: *Nature Genetics* 41.2, pp. 149–155. ISSN: 1546-1718. DOI: 10.1038/ng.295.

Ishizuka, Toru et al. (2006). “Kinetic evaluation of photosensitivity in genetically engineered neurons expressing green algae light-gated channels”. eng. In: *Neuroscience Research* 54.2, pp. 85–94. ISSN: 0168-0102. DOI: 10.1016/j.neures.2005.10.009.

Jia, Hongbo et al. (2011). “In vivo two-photon imaging of sensory-evoked dendritic calcium signals in cortical neurons”. en. In: *Nature Protocols* 6.1, pp. 28–35. ISSN: 1754-2189. DOI: 10.1038/nprot.2010.169. URL: http://www.nature.com/nprot/journal/v6/n1/box/nprot.2010.169_BX1.html?cookies=accepted (visited on 11/18/2016).

Johansen-Berg, H. et al. (2004). “Changes in connectivity profiles define functionally distinct regions in human medial frontal cortex”. en. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.36, pp. 13335–13340. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0403743101. URL: http://www.pnas.org/content/101/36/13335 (visited on 07/03/2017).

Kavalali, Ege T. and Erik M. Jorgensen (2014). “Visualizing presynaptic function”. en. In: *Nature Neuroscience* 17.1, pp. 10–16. ISSN: 1097-6256. DOI: 10.1038/nn.3578. URL: http://www.nature.com/neuro/journal/v17/n1/full/nn.3578.html (visited on 05/26/2017).

Kenall, Amye et al. (2015). “Better reporting for better research: a checklist for reproducibility”. In: *BMC Neuroscience* 16, p. 44. ISSN: 1471-2202. DOI: 10.1186/s12868-015-0177-z. URL: http://dx.doi.org/10.1186/s12868-015-0177-z.

Kenet, T. et al. (2003). “Spontaneously emerging cortical representations of visual attributes”. eng. In: *Nature* 425, pp. 954–6. ISSN: 1476-4687 (Electronic) 0028-0836 (Linking). DOI: 10.1038/nature02078nature02078[pii].

Khakh, Baljit S. and Michael V. Sofroniew (2015). “Diversity of astrocyte functions and phenotypes in neural circuits”. eng. In: *Nature Neuroscience* 18.7, pp. 942–952. ISSN: 1546-1726. DOI: 10.1038/nn.4043.

Kiebel, Stefan J., Marta I. Garrido, and Karl J. Friston (2007). “Dynamic causal modelling of evoked responses: the role of intrinsic connections”. eng. In: *NeuroImage* 36.2, pp. 332–345. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2007.02.046.

Kitzes, J., D. Turek, and F. Deniz (2017). *The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Science*. Oakland, CA: University of California Press. URL: https://www.gitbook.com/book/bids/the-practice-of-reproducible-research/details (visited on 06/26/2017).

Kleinfeld, D. and K.r. Delaney (1996). “Distributed representation of vibrissa movement in the upper layers of somatosensory cortex revealed with voltage-sensitive dyes”. en. In: *The Journal of Comparative Neurology* 375.1, pp. 89–108. ISSN: 1096-9861. DOI: 10.1002/(SICI)1096-9861(19961104)375:1<89::AID-CNE6>3.0.CO;2-K. URL: http://onlinelibrary.wiley.com/doi/10.1002/(SICI)1096-9861(19961104)375:1%3C89::AID-CNE6%3E3.0.CO;2-K/abstract (visited on 12/05/2016).

Knöpfel, Thomas (2012). “Genetically encoded optical indicators for the analysis of neuronal circuits”. eng. In: *Nature Reviews. Neuroscience* 13.10, pp. 687–700. ISSN: 1471-0048. DOI: 10.1038/nrn3293.

Knöpfel, Thomas, Yasir Gallero-Salas, and Chenchen Song (2015). "Genetically encoded voltage indicators for large scale cortical imaging come of age". eng. In: *Current Opinion in Chemical Biology* 27, pp. 75–83. ISSN: 1879-0402. DOI: 10.1016/j.cbpa.2015.06.006.

Konnerth, Arthur and Richard K. Orkand (1986). "Voltage-sensitive dyes measure potential changes in axons and glia of the frog optic nerve". In: *Neuroscience Letters* 66.1, pp. 49–54. ISSN: 0304-3940. DOI: 10.1016/0304-3940(86)90164-3. URL: http://www.sciencedirect.com/science/article/pii/0304394086901643 (visited on 05/26/2017).

Korson, Timothy D. and Vijay K. Vaishnavi (1986). "An Empirical Study of the Effects of Modularity on Program Modifiability". In: *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*. Norwood, NJ, USA: Ablex Publishing Corp., pp. 168–186. ISBN: 978-0-89391-388-5. URL: http://dl.acm.org/citation.cfm?id=21842.28893 (visited on 05/26/2017).

Kötter, Rolf (2007). "Anatomical Concepts of Brain Connectivity". en. In: *Handbook of Brain Connectivity*. Ed. by Viktor K. Jirsa and A. R. McIntosh. Understanding Complex Systems. DOI: 10.1007/978-3-540-71512-2_5. Springer Berlin Heidelberg, pp. 149–167. ISBN: 978-3-540-71462-0 978-3-540-71512-2. URL: http://link.springer.com/chapter/10.1007/978-3-540-71512-2_5 (visited on 05/25/2017).

Kovitz, Benjamin L. (1998). *Practical Software Requirements: A Manual of Content and Style*. English. Greenwich, Conn: Manning Publications. ISBN: 978-1-884777-59-2.

Kozberg, Mariel G. et al. (2016). "Rapid Postnatal Expansion of Neural Networks Occurs in an Environment of Altered Neurovascular and Neurometabolic Coupling". eng. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 36.25, pp. 6704–6717. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.2363-15.2016.

Kuhlman, Sandra J. and Z. Josh Huang (2008). "High-Resolution Labeling and Functional Manipulation of Specific Neuron Types in Mouse Brain by Cre-Activated Viral Gene Expression". In: *PLOS ONE* 3.4, e2005. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0002005. URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0002005 (visited on 08/30/2016).

Li, Kaiming et al. (2009). "Review of methods for functional brain connectivity detection using fMRI". In: *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society* 33.2, pp. 131–139. ISSN: 0895-6111. DOI: 10.1016/j.compmedimag.2008.10.011. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2724763/.

Li, Xiang et al. (2005). "Fast noninvasive activation and inhibition of neural and network activity by vertebrate rhodopsin and green algae channelrhodopsin". en. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.49, pp. 17816–17821. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0509030102. URL: http://www.pnas.org/content/102/49/17816 (visited on 05/25/2017).

Liang, Ruqiang, Gerard Joseph Broussard, and Lin Tian (2015). "Imaging chemical neurotransmission with genetically encoded fluorescent sensors". eng. In: *ACS chemical neuroscience* 6.1, pp. 84–93. ISSN: 1948-7193. DOI: 10.1021/cn500280k.

Lichtman, Jeff W. and Joshua R. Sanes (2008). “Ome sweet ome: what can the genome tell us about the connectome?” eng. In: *Current Opinion in Neurobiology* 18.3, pp. 346–353. ISSN: 0959-4388. DOI: 10.1016/j.conb.2008.08.010.

Lim, Diana H., Jeffrey M. LeDue, Majid H. Mohajerani, et al. (2014). “Optogenetic Mapping after Stroke Reveals Network-Wide Scaling of Functional Connections and Heterogeneous Recovery of the Peri-Infarct”. en. In: *Journal of Neuroscience* 34.49, pp. 16455–16466. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3384-14.2014. URL: http://www.jneurosci.org/content/34/49/16455 (visited on 07/05/2017).

Lim, Diana H., Jeffrey M. LeDue, and Timothy H. Murphy (2015). “Network analysis of mesoscale optical recordings to assess regional, functional connectivity”. In: *Neurophotonics* 2.4. ISSN: 2329-423X. DOI: 10.1117/1.NPh.2.4.041405. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4478876/ (visited on 11/19/2016).

Lim, Diana H., Jeffrey LeDue, et al. (2013). “Optogenetic approaches for functional mouse brain mapping”. In: *Frontiers in Neuroscience* 7. ISSN: 1662-4548. DOI: 10.3389/fnins.2013.00054. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3622058/ (visited on 11/18/2015).

Lim, Diana H., Majid H. Mohajerani, et al. (2012). “In vivo Large-Scale Cortical Mapping Using Channelrhodopsin-2 Stimulation in Transgenic Mice Reveals Asymmetric and Reciprocal Relationships between Cortical Areas”. In: *Frontiers in Neural Circuits* 6. ISSN: 1662-5110. DOI: 10.3389/fncir.2012.00011. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3304170/ (visited on 08/29/2016).

Lin, Michael Z. and Mark J. Schnitzer (2016). “Genetically encoded indicators of neuronal activity”. en. In: *Nature Neuroscience* 19.9, pp. 1142–1153. ISSN: 1097-6256. DOI: 10.1038/nn.4359. URL: https://www.nature.com/neuro/journal/v19/n9/full/nn.4359.html (visited on 05/14/2017).

Liu, Chengbo et al. (2016). “Advances in Imaging Techniques and Genetically Encoded Probes for Photoacoustic Imaging”. In: *Theranostics* 6.13, pp. 2414–2430. ISSN: 1838-7640. DOI: 10.7150/thno.15878. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5118604/ (visited on 11/27/2016).

Lovett-Barron, Matthew et al. (2014). “Dendritic inhibition in the hippocampus supports fear learning”. eng. In: *Science (New York, N.Y.)* 343.6173, pp. 857–863. ISSN: 1095-9203. DOI: 10.1126/science.1247485.

Luo, Liqun, Edward M. Callaway, and Karel Svoboda (2008). “Genetic Dissection of Neural Circuits”. English. In: *Neuron* 57.5, pp. 634–660. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2008.01.002. URL: http://www.cell.com/neuron/abstract/S0896-6273(08)00031-7 (visited on 05/25/2017).

Ma, Liangsuo et al. (2007). “Detecting functional connectivity in the resting brain: a comparison between ICA and CCA”. eng. In: *Magnetic Resonance Imaging* 25.1, pp. 47–56. ISSN: 0730-725X. DOI: 10.1016/j.mri.2006.09.032.

Ma, Ying et al. (2016). “Wide-field optical mapping of neural activity and brain haemodynamics: considerations and novel approaches”. In: *Philosophical Transactions of the Royal Society B:*

Biological Sciences 371.1705. ISSN: 0962-8436. DOI: 10.1098/rstb.2015.0360. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5003860/ (visited on 05/10/2017).

Madisen, L. et al. (2015). "Transgenic mice for intersectional targeting of neural sensors and effectors with high specificity and performance". eng. In: *Neuron* 85, pp. 942–58. ISSN: 1097-4199 (Electronic) 0896-6273 (Linking). DOI: 10.1016/j.neuron.2015.02.022S0896-6273(15)00137-3[pii].

Making Your Code Citable · GitHub Guides (2017). URL: https://guides.github.com/activities/citable-code/ (visited on 06/30/2017).

Malach, R et al. (1993). "Relationship between intrinsic connections and functional architecture revealed by optical imaging and in vivo targeted biocytin injections in primate striate cortex." In: *Proceedings of the National Academy of Sciences of the United States of America* 90.22, pp. 10469–10473. ISSN: 0027-8424. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC47798/ (visited on 05/26/2017).

Malonek, D. and A. Grinvald (1996). "Interactions between electrical activity and cortical microcirculation revealed by imaging spectroscopy: implications for functional brain mapping". eng. In: *Science (New York, N.Y.)* 272.5261, pp. 551–554. ISSN: 0036-8075.

Martin, Robert C (2002). *Agile software development: principles, patterns, and practices*. Prentice Hall.

Marvin, Jonathan S. et al. (2013). "An optimized fluorescent probe for visualizing glutamate neurotransmission". eng. In: *Nature Methods* 10.2, pp. 162–170. ISSN: 1548-7105. DOI: 10.1038/nmeth.2333.

May, Elaine L. and Barbara A. Zimmer (1996). "The Evolutionary Development model for software". In: *Hewlett-Packard Journal* 47.4, p. 39. ISSN: 00181153. URL: http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=9608302686&site=ehost-live&scope=site (visited on 05/26/2017).

McConnell, Steve (2004). *Code complete*. 2nd ed. Redmond, Wash: Microsoft Press. ISBN: 978-0-7356-1967-8.

McIntosh, A. R. et al. (1994). "Network analysis of cortical visual pathways mapped with PET". eng. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 14.2, pp. 655–666. ISSN: 0270-6474.

McKeown, M. J. et al. (1998). "Analysis of fMRI data by blind separation into independent spatial components". eng. In: *Human Brain Mapping* 6.3, pp. 160–188. ISSN: 1065-9471.

Merali, Zeeya (2010). "Computational science: ...Error". eng. In: *Nature* 467.7317, pp. 775–777. ISSN: 1476-4687. DOI: 10.1038/467775a.

Milanovich, Bo (2012). *Python GUI Development with Qt - Introduction - Video 1*. English. URL: https://www.youtube.com/watch?v=mpbWQbkl8_g#t=20m15s (visited on 07/14/2017).

Miyashita, Toshio et al. (2013). "Long-term channelrhodopsin-2 (ChR2) expression can induce abnormal axonal morphology and targeting in cerebral cortex". In: *Frontiers in Neural Circuits* 7, p. 8. DOI: 10.3389/fncir.2013.00008. URL: http://journal.frontiersin.org/article/10.3389/fncir.2013.00008/full (visited on 08/30/2016).

Miyawaki, A. et al. (1997). "Fluorescent indicators for Ca2+ based on green fluorescent proteins and calmodulin". eng. In: *Nature* 388.6645, pp. 882–887. ISSN: 0028-0836. DOI: 10.1038/42264.

Mohajerani, M. H., K. Aminoltejari, and T. H. Murphy (2011). “Targeted mini-strokes produce changes in interhemispheric sensory signal processing that are indicative of disinhibition within minutes”. eng. In: *Proc Natl Acad Sci U S A* 108, E183–91. ISSN: 1091-6490 (Electronic) 0027-8424 (Linking). DOI: 10.1073/pnas.1101914108101914108[pii].

Mohajerani, Majid H., Allen W. Chan, et al. (2013). “Spontaneous cortical activity alternates between motifs defined by regional axonal projections”. In: *Nature neuroscience* 16.10, pp. 1426–1435. ISSN: 1097-6256. DOI: 10.1038/nn.3499. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3928052/ (visited on 08/29/2016).

Mohajerani, Majid H., David A. McVea, et al. (2010). “Mirrored Bilateral Slow-Wave Cortical Activity within Local Circuits Revealed by Fast Bihemispheric Voltage-Sensitive Dye Imaging in Anesthetized and Awake Mice”. en. In: *The Journal of Neuroscience* 30.10, pp. 3745–3751. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.6437-09.2010. URL: http://www.jneurosci.org/content/30/10/3745 (visited on 08/30/2016).

Moran, Rosalyn J. et al. (2011). “An in vivo assay of synaptic function mediating human cognition”. eng. In: *Current biology: CB* 21.15, pp. 1320–1325. ISSN: 1879-0445. DOI: 10.1016/j.cub.2011.06.053.

Moreland, Kenneth (2009). “Diverging Color Maps for Scientific Visualization”. In: *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II*. Ed. by George Bebis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 92–103. ISBN: 978-3-642-10520-3. URL: http://dx.doi.org/10.1007/978-3-642-10520-3_9.

— (2016). “Why We Use Bad Color Maps and What You Can Do About It”. In: *Electronic Imaging* 2016.16, pp. 1–6. DOI: 10.2352/ISSN.2470-1173.2016.16.HVEI-133.

Morin, A. et al. (2012). “Research priorities. Shining light into black boxes”. eng. In: *Science (New York, N.Y.)* 336.6078, pp. 159–160. ISSN: 1095-9203. DOI: 10.1126/science.1218263.

Murphy, Timothy H. et al. (2016). “High-throughput automated home-cage mesoscopic functional imaging of mouse cortex”. en. In: *Nature Communications* 7, p. 11611. DOI: 10.1038/ncomms11611. URL: http://www.nature.com/ncomms/2016/160613/ncomms11611/full/ncomms11611.html (visited on 08/29/2016).

Nagel, Georg et al. (2003). “Channelrhodopsin-2, a directly light-gated cation-selective membrane channel”. en. In: *Proceedings of the National Academy of Sciences* 100.24, pp. 13940–13945. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1936192100. URL: http://www.pnas.org/content/100/24/13940 (visited on 12/08/2015).

Nagel, G. et al. (2005). “Channelrhodopsins: directly light-gated cation channels”. en. In: *Biochemical Society Transactions* 33.4, pp. 863–866. ISSN: 0300-5127, 1470-8752. DOI: 10.1042/BST0330863. URL: http://www.biochemsoctrans.org/content/33/4/863 (visited on 05/26/2017).

Nekrutenko, Anton and James Taylor (2012). “Next-generation sequencing data interpretation: enhancing reproducibility and accessibility”. eng. In: *Nature Reviews. Genetics* 13.9, pp. 667–672. ISSN: 1471-0064. DOI: 10.1038/nrg3305.

Nezafat, Reza, Reza Shadmehr, and Henry Holcomb (2001). “Long-term adaptation to dynamics of reaching movements: a PET study”. In: *Experimental Brain Research* 140.1, pp. 66–76. ISSN: 00144819. DOI: 10.1007/s002210100787. URL: http://link.springer.com/10.1007/s002210100787.

Ng, Minna et al. (2002). “Transmission of olfactory information between three populations of neurons in the antennal lobe of the fly”. eng. In: *Neuron* 36.3, pp. 463–474. ISSN: 0896-6273.

Nguyen, Jeffrey P. et al. (2016). “Whole-brain calcium imaging with cellular resolution in freely behaving Caenorhabditis elegans”. en. In: *Proceedings of the National Academy of Sciences* 113.8, E1074–E1081. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1507110112. URL: http://www.pnas.org/content/113/8/E1074 (visited on 05/26/2017).

Nguyen, Quoc-Thang et al. (2010). “An in vivo biosensor for neurotransmitter release and in situ receptor activity”. en. In: *Nature Neuroscience* 13.1, pp. 127–132. ISSN: 1097-6256. DOI: 10.1038/nn.2469. URL: http://www.nature.com/neuro/journal/v13/n1/abs/nn.2469.html (visited on 05/26/2017).

Nudo, Randolph J. (2013). “Recovery after brain injury: mechanisms and principles”. In: *Frontiers in Human Neuroscience* 7. ISSN: 1662-5161. DOI: 10.3389/fnhum.2013.00887. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3870954/ (visited on 11/18/2015).

Nyberg, Lars et al. (1996). “Network Analysis of Positron Emission Tomography Regional Cerebral Blood Flow Data: Ensemble Inhibition during Episodic Memory Retrieval”. en. In: *Journal of Neuroscience* 16.11, pp. 3753–3759. ISSN: 0270-6474, 1529-2401. URL: http://www.jneurosci.org/content/16/11/3753 (visited on 07/05/2017).

O’Connor, Daniel H. et al. (2010). “Neural Activity in Barrel Cortex Underlying Vibrissa-Based Object Localization in Mice”. English. In: *Neuron* 67.6, pp. 1048–1061. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2010.08.026. URL: http://www.cell.com/neuron/abstract/S0896-6273(10)00636-7 (visited on 05/26/2017).

Oh, Seung Wook et al. (2014). “A mesoscale connectome of the mouse brain”. en. In: *Nature* 508.7495, pp. 207–214. ISSN: 0028-0836. DOI: 10.1038/nature13186. URL: http://www.nature.com/nature/journal/v508/n7495/full/nature13186.html (visited on 08/30/2016).

Okano, Hideyuki, Atsushi Miyawaki, and Kiyoto Kasai (2015). “Brain/MINDS: brain-mapping project in Japan”. en. In: *Phil. Trans. R. Soc. B* 370.1668, p. 20140310. ISSN: 0962-8436, 1471-2970. DOI: 10.1098/rstb.2014.0310. URL: http://rstb.royalsocietypublishing.org/content/370/1668/20140310 (visited on 04/28/2017).

Oller, Sergio (2016). *parmap: map and starmap implementations passing additional arguments and parallelizing if possible*. URL: https://github.com/zeehio/parmap.

Orbach, H. S., L. B. Cohen, and A. Grinvald (1985). “Optical mapping of electrical activity in rat somatosensory and visual cortex”. eng. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 5.7, pp. 1886–1895. ISSN: 0270-6474.

Osten, Pavel and Troy W. Margrie (2013). “Mapping brain circuitry with a light microscope”. In: *Nature methods* 10.6, pp. 515–523. ISSN: 1548-7091. DOI: 10.1038/nmeth.2477. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3982327/ (visited on 11/28/2015).

Patel, Tapan P. et al. (2015). “Automated quantification of neuronal networks and single-cell calcium dynamics using calcium imaging”. In: *Journal of Neuroscience Methods* 243, pp. 26–38. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2015.01.020. URL: http://www.sciencedirect.com/science/article/pii/S0165027015000217 (visited on 12/23/2016).

Pawley, James B. (2006). “Fundamental Limits in Confocal Microscopy”. en. In: *Handbook Of Biological Confocal Microscopy*. Ed. by James B. Pawley. DOI: 10.1007/978-0-387-45524-2_2. Springer US, pp. 20–42. ISBN: 978-0-387-25921-5 978-0-387-45524-2. URL: http://link.springer.com/chapter/10.1007/978-0-387-45524-2_2 (visited on 05/26/2017).

Penfield, W. and E. Boldrey (1937). “Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation”. English. In: *Brain: A Journal of Neurology* 60, pp. 389–443. ISSN: 1460-2156 0006-8950. DOI: 10.1093/brain/60.4.389.

Peterka, Darcy S., Hiroto Takahashi, and Rafael Yuste (2011). “Imaging voltage in neurons”. eng. In: *Neuron* 69.1, pp. 9–21. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2010.12.010.

Peters, Andrew, Simon Chen, and Takaki Komiyama (2014). “Emergence of reproducible spatiotemporal activity during motor learning”. In: *Nature*. ISSN: 0028-0836. DOI: 10.1038/nature13235.

Petersen, Carl C. H. et al. (2003). “Interaction of sensory responses with spontaneous depolarization in layer 2/3 barrel cortex”. en. In: *Proceedings of the National Academy of Sciences* 100.23, pp. 13638–13643. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.2235811100. URL: http://www.pnas.org/content/100/23/13638 (visited on 08/30/2016).

Petersson, K. M. et al. (2000). “Language processing modulated by literacy: a network analysis of verbal repetition in literate and illiterate subjects”. eng. In: *Journal of Cognitive Neuroscience* 12.3, pp. 364–382. ISSN: 0898-929X.

Petreanu, Leopoldo et al. (2007). “Channelrhodopsin-2–assisted circuit mapping of long-range callosal projections”. en. In: *Nature Neuroscience* 10.5, pp. 663–668. ISSN: 1097-6256. DOI: 10.1038/nn1891. URL: http://www.nature.com/neuro/journal/v10/n5/full/nn1891.html (visited on 08/30/2016).

St-Pierre, François, Mariya Chavarha, and Michael Z Lin (2015). “Designs and sensing mechanisms of genetically encoded fluorescent voltage indicators”. In: *Current Opinion in Chemical Biology*. Molecular imaging 27, pp. 31–38. ISSN: 1367-5931. DOI: 10.1016/j.cbpa.2015.05.003. URL: http://www.sciencedirect.com/science/article/pii/S1367593115000459 (visited on 05/26/2017).

St-Pierre, François, Jesse D Marshall, et al. (2014). “High-fidelity optical reporting of neuronal electrical activity with an ultrafast fluorescent voltage sensor”. In: *Nature neuroscience* 17.6, pp. 884–889. ISSN: 1097-6256. DOI: 10.1038/nn.3709. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4494739/ (visited on 05/26/2017).

Podor, Borbala et al. (2015). “Comparison of genetically encoded calcium indicators for monitoring action potentials in mammalian brain by two-photon excitation fluorescence microscopy”. In: *Neurophotonics* 2.2, pp. 021014–021014. ISSN: 2329-423X. DOI: 10.1117/1.NPh.2.2.021014. URL: http://dx.doi.org/10.1117/1.NPh.2.2.021014 (visited on 05/26/2017).

Prinz, Florian, Thomas Schlange, and Khusru Asadullah (2011). “Believe it or not: how much can we rely on published data on potential drug targets?” en. In: *Nature Reviews Drug Discovery* 10.9,

pp. 712–712. ISSN: 1474-1776. DOI: 10.1038/nrd3439-c1. URL: http://www.nature.com/nrd/journal/v10/n9/full/nrd3439-c1.html (visited on 05/26/2017).

PyCharm (2017). URL: https://www.jetbrains.com/pycharm/ (visited on 06/19/2017).

Rajapakse, Jagath C. and Juan Zhou (2007). "Learning effective brain connectivity with dynamic Bayesian networks". eng. In: *NeuroImage* 37.3, pp. 749–760. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2007.06.003.

"Rebooting review" (2015). en. In: *Nature Biotechnology* 33.4, pp. 319–319. ISSN: 1087-0156. DOI: 10.1038/nbt.3202. URL: https://www.nature.com/nbt/journal/v33/n4/full/nbt.3202.html (visited on 06/28/2017).

Rector, David M. et al. (2005). "Spatio-temporal mapping of rat whisker barrels with fast scattered light signals". eng. In: *NeuroImage* 26.2, pp. 619–627. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2005.02.030.

Reddy, B. S. and B. N. Chatterji (1996). "An FFT-based technique for translation, rotation, and scale-invariant image registration". ENG. In: *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society* 5.8, pp. 1266–1271. ISSN: 1057-7149. DOI: 10.1109/83.506761.

Reinhart, Carmen M. and Kenneth S. Rogoff (2010). "Growth in a Time of Debt". In: *American Economic Review* 100.2, pp. 573–578. ISSN: 0002-8282. DOI: 10.1257/aer.100.2.573. URL: https://www.aeaweb.org/articles?id=10.1257/aer.100.2.573 (visited on 07/01/2017).

Rogers, Wm Paul (2001). *Encapsulation is not information hiding*. URL: http://www.javaworld.com/article/2075271/core-java/encapsulation-is-not-information-hiding.html (visited on 05/26/2017).

Rose, Tobias et al. (2014). "Putting a finishing touch on GECIs". In: *Frontiers in Molecular Neuroscience* 7. ISSN: 1662-5099. DOI: 10.3389/fnmol.2014.00088. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4235368/ (visited on 05/26/2017).

S, Spadone et al. (2012). "A K-means multivariate approach for clustering independent components from magnetoencephalographic data." In: *Neuroimage* 62, pp. 1912–1923. URL: https://www.neuroscience.ox.ac.uk/publications/364102 (visited on 07/04/2017).

Sandve, Geir Kjetil et al. (2013). "Ten Simple Rules for Reproducible Computational Research". In: *PLOS Computational Biology* 9.10, e1003285. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003285. URL: http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003285 (visited on 06/26/2017).

Schoenenberger, Philipp, Yan-Ping Zhang Schärer, and Thomas G. Oertner (2011). "Channelrhodopsin as a tool to investigate synaptic transmission and plasticity". en. In: *Experimental Physiology* 96.1, pp. 34–39. ISSN: 1469-445X. DOI: 10.1113/expphysiol.2009.051219. URL: http://onlinelibrary.wiley.com/doi/10.1113/expphysiol.2009.051219/abstract (visited on 08/30/2016).

Schwab, Matthias, Martin Karrenbach, and Jon Claerbout (2000). "Making Scientific Computations Reproducible". In: *Computing in Science and Engg.* 2.6, pp. 61–67. ISSN: 1521-9615. DOI: 10.1109/5992.881708. URL: http://dx.doi.org/10.1109/5992.881708.

SciPy: Scientific Library for Python (2016). https://github.com/scipy/scipy. URL: https://github.com/scipy/scipy (visited on 11/18/2016).

Scott, Benjamin B., Carlos D. Brody, and David W. Tank (2013). “Cellular Resolution Functional Imaging in Behaving Rats Using Voluntary Head Restraint”. In: *Neuron* 80.2, pp. 371–384. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2013.08.002. URL: http://www.sciencedirect.com/science/article/pii/S0896627313007125 (visited on 09/07/2016).

Scott, Nadia A. and Timothy H. Murphy (2012). “Hemodynamic responses evoked by neuronal stimulation via channelrhodopsin-2 can be independent of intracortical glutamatergic synaptic transmission”. eng. In: *PloS One* 7.1, e29859. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0029859.

Sedlmair, M., M. Meyer, and T. Munzner (2012). “Design Study Methodology: Reflections from the Trenches and the Stacks”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12, pp. 2431–2440. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.213.

Seth, Anil K., Adam B. Barrett, and Lionel Barnett (2015). “Granger Causality Analysis in Neuroscience and Neuroimaging”. In: *The Journal of Neuroscience* 35.8, pp. 3293–3297. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.4399-14.2015. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4339347/ (visited on 09/07/2016).

Sheffield, Mark E. J. and Daniel A. Dombeck (2015). “Calcium transient prevalence across the dendritic arbour predicts place field properties”. en. In: *Nature* 517.7533, pp. 200–204. ISSN: 0028-0836. DOI: 10.1038/nature13871. URL: https://www.nature.com/nature/journal/v517/n7533/full/nature13871.html (visited on 05/26/2017).

Shen, Yi et al. (2014). “pHuji, a pH-sensitive red fluorescent protein for imaging of exo- and endocytosis”. en. In: *J Cell Biol* 207.3, pp. 419–432. ISSN: 0021-9525, 1540-8140. DOI: 10.1083/jcb.201404107. URL: http://jcb.rupress.org/content/207/3/419 (visited on 05/26/2017).

Shepherd, Gordon M. G., Thomas A. Pologruto, and Karel Svoboda (2003). “Circuit analysis of experience-dependent plasticity in the developing rat barrel cortex”. eng. In: *Neuron* 38.2, pp. 277–289. ISSN: 0896-6273.

Shibuki, Katsuei, Ryuichi Hishida, et al. (2003). “Dynamic imaging of somatosensory cortical activity in the rat visualized by flavoprotein autofluorescence”. eng. In: *The Journal of Physiology* 549.Pt 3, pp. 919–927. ISSN: 0022-3751. DOI: 10.1113/jphysiol.2003.040709.

Shibuki, Katsuei, Kentaro Ono, et al. (2006). “Endogenous fluorescence imaging of somatosensory cortical activities after discrimination learning in rats”. eng. In: *NeuroImage* 30.3, pp. 735–744. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2005.10.004.

Shih, Chi-Tin et al. (2015). “Connectomics-based analysis of information flow in the Drosophila brain”. eng. In: *Current biology: CB* 25.10, pp. 1249–1258. ISSN: 1879-0445. DOI: 10.1016/j.cub.2015.03.021.

Shoham, Doron et al. (1999). “Imaging Cortical Dynamics at High Spatial and Temporal Resolution with Novel Blue Voltage-Sensitive Dyes”. English. In: *Neuron* 24.4, pp. 791–802. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(00)81027-2. URL: http://www.cell.com/neuron/abstract/S0896-6273(00)81027-2 (visited on 05/26/2017).

Siegel, Friederike and Christian Lohmann (2013). “Probing synaptic function in dendrites with calcium imaging”. eng. In: *Experimental Neurology* 242, pp. 27–32. ISSN: 1090-2430. DOI: 10.1016/j.expneurol.2012.02.007.

Silasi, Gergely and Timothy H. Murphy (2014). “Stroke and the Connectome: How Connectivity Guides Therapeutic Intervention”. In: *Neuron* 83.6, pp. 1354–1368. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2014.08.052. URL: http://www.sciencedirect.com/science/article/pii/S089662731400782X (visited on 11/18/2015).

Silasi, Gergely, Dongsheng Xiao, et al. (2016). “Intact skull chronic windows for mesoscopic wide-field imaging in awake mice”. ENG. In: *Journal of Neuroscience Methods* 267, pp. 141–149. ISSN: 1872-678X. DOI: 10.1016/j.jneumeth.2016.04.012.

Silva, Laysa Mayra Uchôa da et al. (2017). “Measures for brain connectivity analysis: nodes centrality and their invariant patterns”. en. In: *The European Physical Journal Special Topics* 226.10, pp. 2235–2245. ISSN: 1951-6355, 1951-6401. DOI: 10.1140/epjst/e2016-60400-2. URL: https://link.springer.com/article/10.1140/epjst/e2016-60400-2 (visited on 07/05/2017).

Slovin, Hamutal et al. (2002). “Long-Term Voltage-Sensitive Dye Imaging Reveals Cortical Dynamics in Behaving Monkeys”. en. In: *Journal of Neurophysiology* 88.6, pp. 3421–3438. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.00194.2002. URL: http://jn.physiology.org/content/88/6/3421 (visited on 08/29/2016).

Smith, Rachel A. et al. (2002). “The High Cost of Complexity in Experimental Design and Data Analysis: Type I and Type II Error Rates in Multiway ANOVA”. en. In: *Human Communication Research* 28.4, pp. 515–530. ISSN: 1468-2958. DOI: 10.1111/j.1468-2958.2002.tb00821.x. URL: http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2958.2002.tb00821.x/abstract (visited on 05/26/2017).

Soergel, David A. W. (2015). “Rampant software errors may undermine scientific results”. en. In: *F1000Research*. ISSN: 2046-1402. DOI: 10.12688/f1000research.5930.2. URL: http://f1000research.com/articles/3-303/v2.

Sporns, Olaf (2010). *Networks of the Brain*. English. 1 edition. Cambridge, Mass: The MIT Press. ISBN: 978-0-262-01469-4.

Steen, R. Grant (2011). “Retractions in the scientific literature: is the incidence of research fraud increasing?” eng. In: *Journal of Medical Ethics* 37.4, pp. 249–253. ISSN: 1473-4257. DOI: 10.1136/jme.2010.040923.

Stodden, Victoria and Sheila Miguez (2014). “Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research”. en. In: *Journal of Open Research Software* 2.1. ISSN: 2049-9647. DOI: 10.5334/jors.ay. URL: http://openresearchsoftware.metajnl.com/articles/10.5334/jors.ay/ (visited on 06/26/2017).

Suhr, Diana (2006). “The basics of structural equation modeling”. In: *Presented: Irvine, CA, SAS User Group of the Western Region of the United States (WUSS)*.

Sun, Wenzhi et al. (2016). “Thalamus provides layer 4 of primary visual cortex with orientation- and direction-tuned inputs”. eng. In: *Nature Neuroscience* 19.2, pp. 308–315. ISSN: 1546-1726. DOI: 10.1038/nn.4196.

Swanson, Larry W. and Mihail Bota (2010). “Foundational model of structural connectivity in the nervous system with a schema for wiring diagrams, connectome, and basic plan architecture”. en. In:

Proceedings of the National Academy of Sciences 107.48, pp. 20610–20617. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1015128107. URL: http://www.pnas.org/content/107/48/20610 (visited on 05/26/2017).

Swanson, Larry W. and Jeff W. Lichtman (2016). "From Cajal to Connectome and Beyond". In: *Annual Review of Neuroscience* 39.1, pp. 197–216. DOI: 10.1146/annurev-neuro-071714-033954. URL: http://dx.doi.org/10.1146/annurev-neuro-071714-033954 (visited on 05/08/2017).

Swanson, Larry W., Olaf Sporns, and Joel D. Hahn (2016). "Network architecture of the cerebral nuclei (basal ganglia) association and commissural connectome". eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 113.40, E5972–E5981. ISSN: 1091-6490. DOI: 10.1073/pnas.1613184113.

Swindale, Nicholas V. and Martin A. Spacek (2014). "Spike sorting for polytrodes: a divide and conquer approach". eng. In: *Frontiers in Systems Neuroscience* 8, p. 6. ISSN: 1662-5137. DOI: 10.3389/fnsys.2014.00006.

Tabares, Lucia et al. (2007). "Monitoring Synaptic Function at the Neuromuscular Junction of a Mouse Expressing SynaptopHluorin". en. In: *Journal of Neuroscience* 27.20, pp. 5422–5430. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0670-07.2007. URL: http://www.jneurosci.org/content/27/20/5422 (visited on 05/26/2017).

Takerkart, Sylvain et al. (2014). "Vobi One: a data processing software package for functional optical imaging". In: *Frontiers in Neuroscience* 8. ISSN: 1662-4548. DOI: 10.3389/fnins.2014.00002. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3901006/ (visited on 12/23/2016).

Tanigawa, Hisashi, Haidong D. Lu, and Anna W. Roe (2010). "Functional organization for color and orientation in macaque V4". en. In: *Nature Neuroscience* 13.12, pp. 1542–1548. ISSN: 1097-6256. DOI: 10.1038/nn.2676. URL: http://www.nature.com/neuro/journal/v13/n12/full/nn.2676.html (visited on 11/27/2016).

Theis, Lucas et al. (2016). "Benchmarking Spike Rate Inference in Population Calcium Imaging". In: *Neuron* 90.3, pp. 471–482. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2016.04.014. URL: http://www.sciencedirect.com/science/article/pii/S0896627316300733 (visited on 05/26/2017).

Tinati, Ramine, Markus Luczak-Roesch, Elena Simperl, and Wendy Hall (2016). "Because Science is Awesome: Studying Participation in a Citizen Science Game". In: *Proceedings of the 8th ACM Conference on Web Science*. WebSci '16. New York, NY, USA: ACM, pp. 45–54. ISBN: 978-1-4503-4208-7. DOI: 10.1145/2908131.2908151. URL: http://doi.acm.org/10.1145/2908131.2908151 (visited on 05/26/2017).

Tinati, Ramine, Markus Luczak-Roesch, Elena Simperl, Nigel Shadbolt, et al. (2015). "'/Command' and Conquer: Analysing Discussion in a Citizen Science Game". In: *Proceedings of the ACM Web Science Conference*. WebSci '15. New York, NY, USA: ACM, 26:1–26:10. ISBN: 978-1-4503-3672-7. DOI: 10.1145/2786451.2786455. URL: http://doi.acm.org/10.1145/2786451.2786455 (visited on 05/26/2017).

Ting, Jonathan T. and Guoping Feng (2013). "Development of transgenic animals for optogenetic manipulation of mammalian nervous system function: Progress and prospects for behavioral

neuroscience". In: *Behavioural Brain Research*. Optogenetics 255, pp. 3–18. ISSN: 0166-4328. DOI: 10.1016/j.bbr.2013.02.037. URL: http://www.sciencedirect.com/science/article/pii/S0166432813001198 (visited on 08/30/2016).

Tyc, Matej and Christoph Gohlke (2016). *imreg_dft - Image registration using discrete Fourier transform.* https://github.com/matejak/imreg_dft. URL: https://github.com/matejak/imreg_dft (visited on 11/19/2016).

Vanni, Matthieu P., Allen W. Chan, et al. (2017). "Mesoscale mapping of mouse cortex reveals frequency-dependent cycling between distinct macroscale functional modules". en. In: *Journal of Neuroscience*, pp. 3560–16. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3560-16.2017. URL: http://www.jneurosci.org/content/early/2017/07/03/JNEUROSCI.3560-16.2017 (visited on 07/04/2017).

Vanni, Matthieu P. and Timothy H. Murphy (2014). "Mesoscale transcranial spontaneous activity mapping in GCaMP3 transgenic mice reveals extensive reciprocal connections between areas of somatomotor cortex". ENG. In: *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience* 34.48, pp. 15931–15946. ISSN: 1529-2401. DOI: 10.1523/JNEUROSCI.1818-14.2014.

Vanvinckenroye, Amaury et al. (2016). "Eyes Open on Sleep and Wake: In Vivo to In Silico Neural Networks". en. In: *Neural Plasticity* 2016, e1478684. ISSN: 2090-5904. DOI: 10.1155/2016/1478684. URL: http://www.hindawi.com/journals/np/2016/1478684/abs/ (visited on 11/27/2016).

Venkatachalam, Vivek et al. (2016). "Pan-neuronal imaging in roaming Caenorhabditis elegans". eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 113.8, E1082–1088. ISSN: 1091-6490. DOI: 10.1073/pnas.1507109113.

Vesterinen, Hanna M. et al. (2011). "Systematic survey of the design, statistical analysis, and reporting of studies published in the 2008 volume of the Journal of Cerebral Blood Flow and Metabolism". eng. In: *Journal of Cerebral Blood Flow and Metabolism: Official Journal of the International Society of Cerebral Blood Flow and Metabolism* 31.4, pp. 1064–1072. ISSN: 1559-7016. DOI: 10.1038/jcbfm.2010.217.

Walt, S. van der, S. C. Colbert, and G. Varoquaux (2011). "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2, pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.

Wekselblatt, Joseph B. et al. (2016). "Large-scale imaging of cortical dynamics during sensory perception and behavior". eng. In: *Journal of Neurophysiology* 115.6, pp. 2852–2866. ISSN: 1522-1598. DOI: 10.1152/jn.01056.2015.

White, Brian R. et al. (2011). "Imaging of functional connectivity in the mouse brain". eng. In: *PloS One* 6.1, e16322. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0016322.

Wilson, Greg et al. (2014). "Best practices for scientific computing". eng. In: *PLoS biology* 12.1, e1001745. ISSN: 1545-7885. DOI: 10.1371/journal.pbio.1001745.

Wilt, Brian A., James E. Fitzgerald, and Mark J. Schnitzer (2013). "Photon shot noise limits on optical detection of neuronal spikes and estimation of spike timing". eng. In: *Biophysical Journal* 104.1, pp. 51–62. ISSN: 1542-0086. DOI: 10.1016/j.bpj.2012.07.058.

Wiltschko, Alexander B. et al. (2015). “Mapping Sub-Second Structure in Mouse Behavior”. English. In: *Neuron* 88.6, pp. 1121–1135. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2015.11.031. URL: http://www.cell.com/neuron/abstract/S0896-6273(15)01037-5 (visited on 05/26/2017).

Windischberger, Christian et al. (2003). “Fuzzy cluster analysis of high-field functional MRI data”. eng. In: *Artificial Intelligence in Medicine* 29.3, pp. 203–223. ISSN: 0933-3657.

Winship, Ian R. and Timothy H. Murphy (2008). “In Vivo Calcium Imaging Reveals Functional Rewiring of Single Somatosensory Neurons after Stroke”. en. In: *Journal of Neuroscience* 28.26, pp. 6592–6606. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.0622-08.2008. URL: http://www.jneurosci.org/content/28/26/6592 (visited on 01/12/2017).

Wolff, Tanya, Nirmala A. Iyer, and Gerald M. Rubin (2015). “Neuroarchitecture and neuroanatomy of the Drosophila central complex: A GAL4-based dissection of protocerebral bridge neurons and circuits”. eng. In: *The Journal of Comparative Neurology* 523.7, pp. 997–1037. ISSN: 1096-9861. DOI: 10.1002/cne.23705.

Xerri, Christian (2012). “Plasticity of cortical maps: multiple triggers for adaptive reorganization following brain damage and spinal cord injury”. eng. In: *The Neuroscientist: A Review Journal Bringing Neurobiology, Neurology and Psychiatry* 18.2, pp. 133–148. ISSN: 1089-4098. DOI: 10.1177/1073858410397894.

Xiao, Dongsheng et al. (2017). “Mapping cortical mesoscopic networks of single spiking cortical or sub-cortical neurons”. en. In: *eLife* 6, e19976. ISSN: 2050-084X. DOI: 10.7554/eLife.19976. URL: https://elifesciences.org/articles/19976 (visited on 07/13/2017).

Xie, Yicheng et al. (2016). “Resolution of High-Frequency Mesoscale Intracortical Maps Using the Genetically Encoded Glutamate Sensor iGluSnFR”. en. In: *The Journal of Neuroscience* 36.4, pp. 1261–1272. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.2744-15.2016. URL: http://www.jneurosci.org/content/36/4/1261 (visited on 08/29/2016).

Xiong, J. et al. (1999). “Interregional connectivity to primary motor cortex revealed using MRI resting state images”. eng. In: *Human Brain Mapping* 8.2-3, pp. 151–156. ISSN: 1065-9471.

Zhang, Feng, Alexander M. Aravanis, et al. (2007). “Circuit-breakers: optical technologies for probing neural signals and systems”. eng. In: *Nature Reviews. Neuroscience* 8.8, pp. 577–581. ISSN: 1471-003X. DOI: 10.1038/nrn2192.

Zhang, Feng, Li-Ping Wang, et al. (2006). “Channelrhodopsin-2 and optical control of excitable cells”. en. In: *Nature Methods* 3.10, pp. 785–792. ISSN: 1548-7091. DOI: 10.1038/nmeth936. URL: http://www.nature.com/nmeth/journal/v3/n10/full/nmeth936.html (visited on 08/30/2016).

Zheng, Xuebin and Jagath C. Rajapakse (2006). “Learning functional structure from fMR images”. eng. In: *NeuroImage* 31.4, pp. 1601–1613. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2006.01.031.

Zingg, Brian et al. (2014). “Neural Networks of the Mouse Neocortex”. In: *Cell* 156.5, pp. 1096–1111. ISSN: 0092-8674. DOI: 10.1016/j.cell.2014.02.023. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4169118/ (visited on 05/25/2017).

Ziv, Yaniv et al. (2013). “Long-term dynamics of CA1 hippocampal place codes”. en. In: *Nature Neuroscience* 16.3, pp. 264–266. ISSN: 1097-6256. DOI: 10.1038/nn.3329. URL: http://www.nature.com/neuro/journal/v16/n3/full/nn.3329.html (visited on 11/09/2015).

Zou, Peng et al. (2014). “Bright and fast multicoloured voltage reporters via electrochromic FRET”. en. In: *Nature Communications* 5, p. 4625. ISSN: 2041-1723. DOI: 10.1038/ncomms5625. URL: http://www.nature.com/ncomms/2014/140813/ncomms5625/full/ncomms5625.html (visited on 05/26/2017).

Appendices

Appendix A

Available Plugins

A.1 Average

This plugin averages activity in a selected image stack into a single frame

A.2 Alignment

This plugin makes use of the Python image registration utility imreg_dft to implement a means of optimizing translation, rotation and scale variation between two images (Reddy and Chatterji 1996; Tyc and Gohlke 2016). The user can decide whether rotation and scale is also accounted for. The user can compute a reference frame that is the x'th to y'th frame (where x,y are user-defined) of a single image stack averaged. A fast-fourier transform technique is subsequently used to translate, rotate and scale the z'th frame in each selected image (where z is user-defined) to align this frame to the reference frame. The translation, rotation and scale required for this transformation is then applied to all frames in that image stack. This plugin therefore assumes there is negligible movement within a single image stack.

A.3 Calculate df over f0

This plugin computes the average across all frames to establish a baseline. The change in fluorescence from this averaged baseline for each frame can be computed ($\Delta F/F_0$). This processing step results in data more robust against slow drifting of the baseline signal and fast oscillatory noise due to tissue pulsation, thus ensuring the signal detected more accurately represents brain activity (Jia et al. 2011)(i.e. calcium, glutamate or voltage transients).

A.4 Channel Math

Arithmetic division, multiplication, addition or subtraction can be performed frame-by-frame between two selected files. An important use case covered by this plugin is potential hemodynamic artifact correction. If diffuse-reflectance isosbestic point signals are used that can reflect blood volume changes then

contamination removal can be performed by dividing the fluorescence F/F_0 (typically green epi-fluorescence) by the reflectance F/F_0. Reflectance F/F_0 is typically sourced from green light or in some cases a blue reflection image also near an isosbestic point, given intrinsic hemodynamic blood volume signals can be measured with either (Ron D. Frostig and Chen-Bee 2009; Y. Ma et al. 2016; Xiao et al. 2017).The same approximate result can be achieved by subtracting reflectance $\Delta F/F_0$ from fluorescence $\Delta F/F_0$.
Another important use of this plugin is that two identical F/F_0 image stacks can be multiplied and then the result averaged into a single image stack using the average plugin (see A.1) to create a root mean square to assess the overall $\Delta F/F_0$ activity in regions (Chan et al. 2015).

A.5 Concatenation

This plugin concatenates selected image stacks into one single stack in the order stacks are selected.

A.6 Correlation matrix

To generate a correlation matrix the pixel values in an ROI are averaged for each stack and the resultant one-dimensional array is compared with the arrays from other ROIs. Pearson correlation coefficients are computed for each selected ROI-ROI pair. The averaged correlation value for each ROI-ROI pair across image stacks is outputted in the final matrix. The standard deviation of the correlation values for each ROI-ROI pair is likewise computed and included in the final output. Ultimately the resultant matrix shows how correlated activity between brain regions are (Chan et al. 2015). Values from the matrix can be saved to a csv file in the project directory.

A.7 Create ROIs

MBE was originally inspired by a BMDanalyse, a program designed for the regional analysis of bone mass density through interactive visualizations (Hogg 2013). The application makes use of PyQtGraph, a pure Python library that leverages numpy for computation and Qt's GraphicsView framework for fast display (Campagnola 2016; Walt, Colbert, and Varoquaux 2011). It provides well-written classes for ROI-based data slicing on top of the pyqtgraph framework as well as a GUI written in PyQt4, a popular industry-standard framework that supports multiprocessing. All of these tools were adapted from Hogg's original program and adapted for use in this plugin for the generation of polygon ROIs and cropping to selected polygon ROIs across a stack of images.

A.8 Crop Border

Crops a user-defined percentage of the total width of an image stack from all sides of an image stack. This is typically used to crop image stacks so that only brain matter and blood veins are in view, so that only these features are used to align one image stack to another, which can greatly improve alignment accuracy.

A.9 Empty Plugin

This is a template plugin that developers can use. It provides a list of image stacks that can be selected to provide an interactable pyqtgraph view of the first frame of the selected image stack along with an x-y coordinate frame, a drop-down list for filtering image stacks according to the last processing step it went through as well as a video player to view all frames in an image stack with a slider once a list item is double-clicked. Finally a button provided is connected with an empty function that a developer could expand.

A.10 Evoked Average

The response to sensory stimulation (light flashes) can be recorded and averaged to map visual cortical areas (Vanni and Timothy H. Murphy 2014). These sensory evoked averages are referred to as motifs (Majid H. Mohajerani, Chan, et al. 2013). The propensity for spatial-temporal activity motifs to repeat in a set of spontaneous activity can then be assessed (Majid H. Mohajerani, Chan, et al. 2013). This plugin averages selected image stacks across image stacks to create an averaged image stack. This operation is typically used to generate the aforementioned sensory or behavior evoked average motifs.

A.11 Export Files

Image stacks can be exported to .tif, .raw or .mp4 formats. Further file format support could be implemented within this plugin.

A.12 Global Signal Regression

GSR can optionally be applied to remove spontaneous fluctuations common to the whole brain using a general linear model. GSR has been shown to facilitate the detection of localized neuronal signals and improve the specificity of functional connectivity analysis (Fox, D. Zhang, et al. 2009).

A.13 Import CSV ROI Coordinates

Please refer to the README for instructions regarding how to structure coordinates to be used by this plugin: (Haupt 2017).
Square ROIs are drawn at brain locations using coordinates specified by the user via a .csv or .txt file which is loaded. Each ROI additionally has its own custom size. All ROI's loaded are saved to the project directory as a .ROI file to be used across plugins.

A.14 Import Image Stacks

.tif and .raw are fully supported with .tiff reading handled by tifffile (Gohlke 2014). All files are converted to Python numpy arrays (.npy) upon import. .npy formats can also be imported directly. Support for other formats can be implemented within this plugin.

1) ROI Name	2) Length	3) X Coordinate	4) Y Coordinate
L-V1	1	-2516.8	-4267.8
L-BC	1	-4300	-760
L-HL	1	-1694.2	-1145.7
L-M1	1	-1500	2000
L-M2	1	-870.02	1420.5
L-RS	1	-620.43	-2885.8
L-AC	1	-260	270
R-V1	1	2516.8	-4267.8
R-BC	1	4300	-760
R-HL	1	1694.2	-1145.7
R-M1	1	1500	2000
R-M2	1	870.02	1420.5
R-RS	1	620.43	-2885.8
R-AC	1	260	270

Table A.1: Table of the CSV file with coordinates in microns the same as those used in Fig. 2.3b. This table is also identical to the CSV file used to to specify the ROIs used in Fig. 2.7a. The length column here specifies that all ROIs are square single-pixel wide ROIs. The csv includes ROIs for the AC, V1, M2, BC, RS, M1 and the HL for each hemisphere (left - L and right - R). Coordinates were adapted from the Allen Mouse Brain Connectivity Atlas (*Allen Mouse Brain Connectivity Atlas* 2012; Oh et al. 2014). The position of seeds for BC and M1 were adjusted to maximize the remote correlation between them. Their positions are still within the general region of motor and barrel cortex (*Allen Mouse Brain Connectivity Atlas* 2012). We previously mapped functional and anatomical coordinates of transgenic mice using sensory stimulation in combination with in vivo large-scale cortical mapping using CHR2 stimulation to confirm the coordinates above (Lim, Majid H. Mohajerani, et al. 2012; Majid H. Mohajerani, Chan, et al. 2013).

A.15 Plot ROI Activity

ROI activity across stacks can be plotted for all selected ROI's. This plugin opens an interactive pyqtgraph GraphicsWindow where the graph of activity can be manipulated (e.g. zoomed in on) before being exported to an image file, scalable vector graphic, matplotlib window, CSV or HDF5 (Fig. 2.5b shows data that has been exported from the plugin to a CSV. The graph was subsequently made using Plotly (Inc 2015), an online data visualization and analytics tool). (Campagnola 2016).

A.16 SPC Map

The user can click a single pixel called the seed. Pearson correlation zero lag is used to generate a color map showing how brain activity over time at each pixel correlates with brain activity at the seed (Vanni and Timothy H. Murphy 2014). SPC maps thus reveal brain regions displaying synchronous activity.
The user can also use a list of defined seeds with defined locations from the .csv or .txt file loaded via the "Import CSV ROI Coordinates" plugin. This will either generate SPC maps in a separate windows for each selected seed or simply save SPC maps for all seeds across all selected image stacks to the project directory

as .jpeg files and as .npy files that store all pixel values.

A.17 Set Coordinate System

The origin, used as a reference point for user-specified coordinates, can be specified at this plugin. The user can select an origin per image stack. This value is stored for each individual file's JSON parameter. An averaged origin across selected files' origins can then be generated to set the origin for the entire project. In the typical use-case, X and Y are centered on the anatomical landmark bregma.

A.18 Shift Across Projects

MBE works with a single project wherein a JSON file defines all plugins and data in the project directory used by MBE for that project. A single project has a single origin across all files. In our example the origin was set to be the location of bregma. As such, using image stacks across different mice is not feasible as the algorithm used by the alignment plugin is strongly influenced by blood vessels. It is much more feasible to shift all stacks from other animals to match the origin of the current project. This is what this plugin achieves, allowing the user to shift and then immediately import specific selected image stacks from multiple projects. This is needed, for example, in the generation of averaged correlation matrices across many image stacks from different mice.

A.19 Standard Deviation Map

The user selects a maximum value for standard deviation and the plugin computes a standard deviation map showing how much brain activity varies over time at each pixel. The max value is taken as the upper limit of the map scale. The user can also specify a maximum standard deviation, limiting the upper bound of the scale.

A.20 Temporal Filter

A temporal filter can be applied to all selected image stacks, with the user specifying the passband of allowed brain activity signal. This increases the signal-to-noise ratio by removing noise such as cardiac factors (Vanni and Timothy H. Murphy 2014; Carandini et al. 2015). Presently the only filtering algorithms MBE provides is Chebyshev (Type I digital and analog filter design, order = 4, maximum allowable ripple in passband = 0.1) with high and low passbands and frame rate user-specified. This linear MATLAB-style infinite impulse response filter is subsequently applied once forward and once backwards across stacks for each selected stack (*SciPy: Scientific Library for Python* 2016). Other filtering algorithms such as butterworth - also available from the same Python package as the Chebyshev filter - could easily be implemented into this plugin.

A.21 Trimming

The user can define how many frames can be discarded from the start and end of all selected image stacks. This plugin is primarily for cleaning the initial data (e.g. removing movement artifacts at the start and end of a recording)

A.22 Unsharp Filter

An unsharp filter subtracts an "unsharp," or smoothed, version of an image from the original image. This outputs an image with enhanced edges (Haralick and Shapiro 1992).
Each frame in the image stack is first smoothed. The size of the mean filter kernel is selected and each frame in the selected image stacks are convolved with the given kernel. This smooths each frame by reducing the variation between one pixel and the next with kernel size controlling the magnitude of smoothing. This filtered image stack is subtracted from the original image stack frame by frame thereby prominently highlighting (i.e. sharpening) features in the stack that are a particular size, relative to the kernel size.
Unsharp filtering is often used to highlight blood vessels. A frame from the sharpened stack is then used as the reference frame for alignment (see A.2) such that frames across other image stacks are aligned based on the location of blood vessels in the reference frame.

Appendix B

Mesoscale Brain Explorer Version Reference

The following versions of MBE are referred to in the thesis:

- 0.7.10 - This is the version that was released upon the submission of the manuscript: “Mesoscale brain explorer, a flexible python-based image analysis and visualization tool”. As this manuscript and chapter 2 are identical, this reference is exclusively referenced there.
- 0.8.0 - Represents a major overhaul since the 0.7 releases, and the 0.7.10 release that coincided with the manuscript submission. The visualization windows with docks (see Sect. 3.2.2) have their final design framework implemented whereas before they were fragile, pipeline automation is reworked to support multiple simulataneous pipelines, allowing for a single pipeline where some processing steps are performed on one set of data, and other processing steps on another set (or the same set). At the time of writing, 0.8 releases represent the current development of MBE.
- 1.0.0 - This version is not finished and is the version that would denote that MBE has met industry-grade standards for public release. This would include the near elimination of all bugs. Releases prior to this release are termed "pre-releases" as bugs are more common. Bugs in versions 0.7-0.8 typically do not functionally break the application, but they can cause annoyances for which workarounds need to be performed. As such, industry-grade standards have not been met and MBE remains at the pre-release stage of development, despite being fully functional.

Appendix C

Custom Plugin Tutorial

C.1 Overview

This step-by-step tutorial will lead the reader through the steps required for adding a custom plugin to MBE. In this tutorial the plugin to be developed imports a list of numbers and adds each number to an image stacks such that all pixels in each image stack have been added by a number. i.e If the list consists of integers x,y,z and we have image stacks A,B,C then the resulting image stacks are A+x,B+y,C+z. We'll integrate the plugin with MBE's architecture so that an automated processing pipeline can be constructed that includes the custom addition plugin. This plugin is merely an example and serves no analytic purpose. It is intended to showcase a common example where external data (i.e. addition data) are imported and used with existing data (i.e. the image stack) already imported.

Note that this tutorial assumes at least undergraduate level Python programming knowledge. You should understand types, loops, breaks, dictionaries, exceptions, functions as well as object oriented programming - classes and how to use them. Any Integrated Development Environment (IDE) should be fine and this tutorial assumes you already have an IDE set-up. We highly recommend Pycharm (*PyCharm* 2017) if you do not have a preferred IDE. This tutorial assumes you are generally familiar with MBE's user interface (see Haupt et al. (2017)). Lastly, this tutorial also assumes you have already followed the For Developers section in MBE's README (See Haupt (2017)) to set up an environment for yourself where you are ready to modify the code.

This tutorial is written assuming no knowledge of Python application development and desktop development concepts will be introduced as is needed. However, thorough knowledge of PyQt certainly doesn't hurt and can greatly facilitate what you can do with your own custom plugin. We recommend Milanovich (2012) and/or Bodnar (2015) for a tutorial on Python GUI development. You should be able to complete them in about a week or two.

Full code used in this tutorial can be found at Haupt (2017).

C.2 Part 1 - Addition Plugin Base Functionality

1. Create a new Python file under src/plugins where all plugin scripts are kept. We named the Python file "addition_example" for this tutorial.

2. Copy all the code from "src/plugins/template_plugin.py" to your new empty Python file. This code is an empty plugin that can be used as a template.

3. Scroll down to the MyPlugin class in "addition_example.py". This class has an attribute:

   ```
   name = "Empty Plugin"
   ```

   Change this to:

   ```
   name = "Addition Example"
   ```

4. Run MBE by running pipegui.py. Go to Configure Pipeline. You should see "Addition Example" listed there. You can add it to the pipeline list in the configure pipeline window and it should appear in the plugin list in the pipeline pane (3.2).

5. Go to the todo comment *Define global attributes and UI components here*.

6. Erase the two lines defining *main_button* and *example_sb* and replace them with:

   ```
   self.add_btn = QPushButton('Perform Addition')
   self.import_additions_btn = QPushButton('Import List of Additions')
   self.add_list = QListWidget()
   ```

   Here we are defining two buttons and one list.

7. Go to the todo marked *setup UI component layout and properties here*. This is where you take your defined UI components and decide where and how to place them in the UI. The function where this occurs is called *setup_ui*.

8. Replace the following five lines of code:

   ```
   self.example_sb.setMaximum(1000)
   self.vbox.addWidget(QLabel(self.Labels.example_sb_label))
   self.vbox.addWidget(self.example_sb)
   self.vbox.addStretch()
   self.vbox.addWidget(self.main_button)
   ```

   with:

   ```
   self.vbox.addWidget(QLabel("Select Additions"))
   self.vbox.addWidget(self.add_list)
   ```

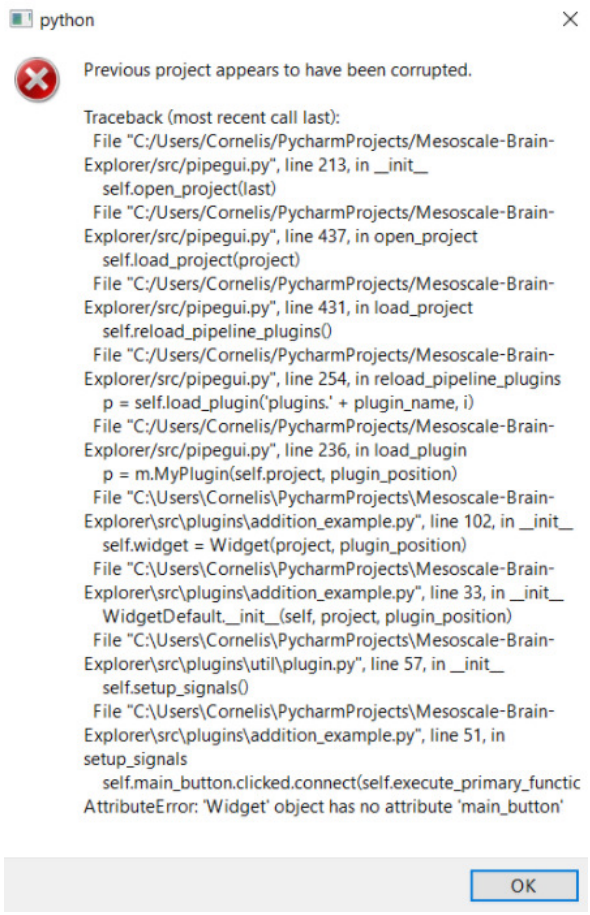


Figure C.1: MBE error message.

```
self.vbox.addWidget(self.import_additions_btn)
self.vbox.addStretch()
self.vbox.addWidget(self.add_btn)
```

The original 5 lines added a label, then a spinbox that has its maximum value set to 1000, then a stretch (blank space) and then the main button. Note here that the order in which you add components is the order (from top to bottom) they are added to the plugin. This is controlled via the inherited *QVBoxLayout* defined as *vbox* which lines up components vertically. The new 5 lines add a label, then the list which will contain the additions, then a button to import those additions, then a stretch, and finally the addition button.

9. Run the modified code. You should be confronted with Fig. C.1). If you press OK MBE will still run. However if you try to navigate to your Addition plugin you'll be confronted with the same traceback in the terminal. Firstly, despite one plugin having an error, the rest of MBE is still fully functional. Secondly, look at the very last line in the traceback which tells us the line where the error starts:

```
File "C:<location of src>\plugins\addition_example.py", line 51, in
    setup_signals
   self.main_button.clicked.connect(self.execute_primary_function)
AttributeError: 'Widget' object has no attribute 'main_button'
```

10. Go to line your traceback outputs (here 51) in *addition_example.py* in the *setup_signals* function. Note that if you are using Pycharm you can right-click on the right margin to show line numbers. At the line in question you'll notice the *main_button* variable originally defined in the template is still being used. You deleted this in step 6.

11. For now simply replace all instances of *main_button* with *add_btn*. If you are using Pycharm you can highlight the variable *main_button* and click ctrl+R to bring up the find + replace interface to use "Replace All."

12. Run MBE. It will still show an error message, but this time indicate that *'Widget' object has no attribute 'example_sb.'*

13. Comment out all of lines where 'example_sb' appears. Do not delete these lines as we will refer to them later.

14. Rerun MBE and navigate to your plugin. You should notice that your UI components are now situated in the Plugin UI Interface (Figs. 3.2, C.2):

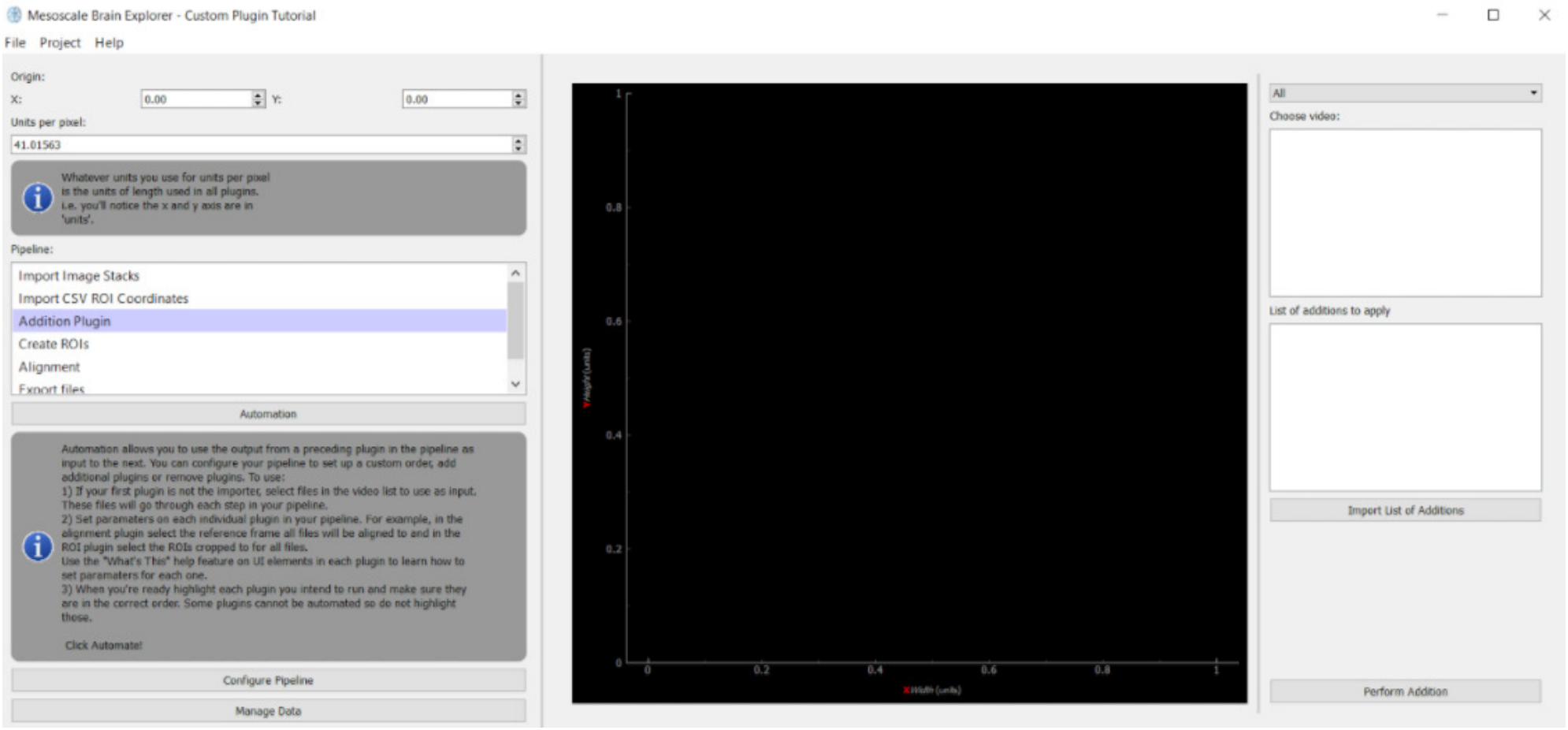


Figure C.2: MBE with addition plugin UI components.

15. The interface does not respond to interaction yet. Go to the *Setup signals (i.e. what ui components do) here* todo comment. Add the following line to the *setup_signals* function:

```
self.import_additions_btn.clicked.connect(self.import_additions)
```

This takes the *import_additions_btn* and connects the action *clicked* (which is a mouse click) with the function *import_additions*. Note that you can of course connect to a function outside of the *Widget* class by replacing *self* with the appropriate object.

16. Add this *import_additions* function just before the *execute_primary_function* function:

```
def import_additions(self):
  text_file_path = QFileDialog.getOpenFileName(
     self, 'Load images', QSettings().value('last_load_text_path'),
     'Video files (*.csv *.txt)')
  if not text_file_path:
     return
  QSettings().setValue('last_load_text_path',
     os.path.dirname(text_file_path))
  copyfile(text_file_path, os.path.join(self.project.path,
     os.path.basename(text_file_path)))
  text_file_path = os.path.join(self.project.path, text_file_path)


  add_list = []
  with open(text_file_path, 'rt', encoding='ascii') as csvfile:
    add_list_it = csv.reader(csvfile)
    for row in add_list_it:
      add_list = add_list + row
  self.add_list.addItems(add_list)
```

Note that you will need:

```
import os, csv
from PyQt4.QtCore import QSettings
from shutil import copyfile
```

When you click the "Import List of Additions" button the above function is triggered. The first line loads up a *QFileDialog* that lets the user pick the file and returns the file's location and assigns it to the variable *text_file_path*. *QSettings()* is used to load the last load location so that the the *QFileDialog* opens to the same location and the user doesn't have to navigate to a particular folder over and over again. If the user did not select a file then the function exits. Finally, *copyfile* is used to copy the file to the project directory, *self.project.path*, and the new location of the file in the project directory is set as the *text_file_path*.

Subsequently, *csvfile* loads the file and assigns each item to the list variable *add_list* which is then added to the UI component *self.add_list*.

17. Create a .csv or .txt with a list of numbers in it, each in its own line. For this tutorial we simply used:

```
3
50
23
```

Run MBE and load this file in your custom plugin and these values should be inserted into the *QListWidget* you've just coded as in Fig. C.3:

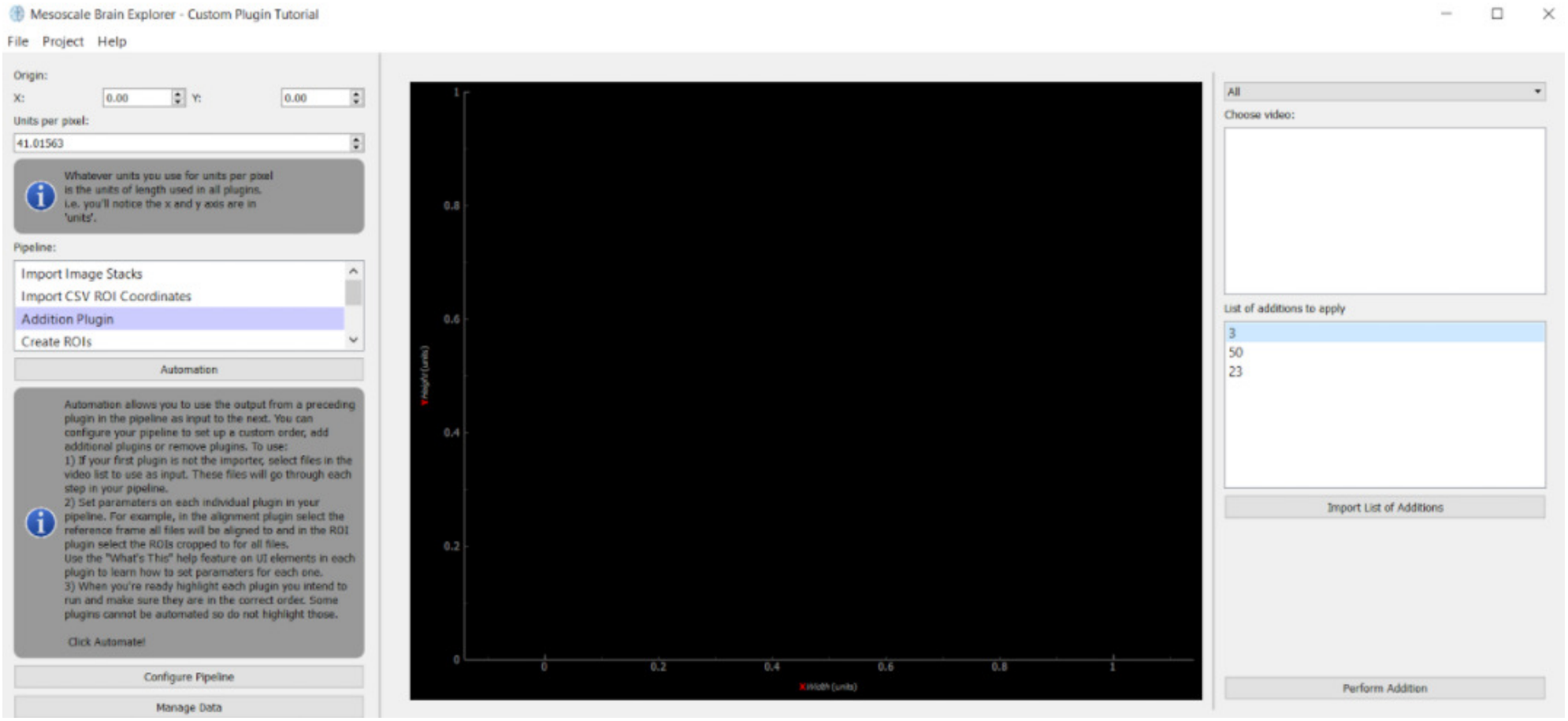


Figure C.3: Addition plugin with three .csv numbers imported.

18. We can use the *qtutil* module to bring up a popup window. This could be used to inform the user if the number of additions are too few or too many for the number of image stacks. Delete the line that makes use of qtutil that you commented out in step 13 (right above the *insert functionality here* todo) and replace it with:

```
if not len(self.add_list.selectedItems()) == len(selected_videos):
  qtutil.info('Select the same number of image stacks and additions to
     apply')
  Return
```

If the number of selected videos and the number of selected items in *add_list* are not the same the function exits and a popup message appears.

19. If the user selects numbers from *add_list* then the GUI component must be made to accommodate multiple selections which it does not do by default. To enable extended selection add the following line in *setup_ui*:

```
self.add_list.setSelectionMode(QAbstractItemView.ExtendedSelection)
```

20. Go to the *insert functionality here* todo and add this below it:

```
if not len(self.add_list.selectedIndexes()) == len(selected_videos):
  qtutil.info('Select the same number of image stacks and additions to
     apply')
  return
for add_num_item, video_path in zip(self.add_list.selectedItems(),
    selected_videos):
  frames = file_io.load_file(video_path)
  frames = frames + int(add_num_item.text())
  path = pfs.save_project(video_path, self.project, frames,
     'custom-addition', 'video')
```

First the function checks to make sure that the number of selected additions is equal to the number of selected videos. *qtutil* (which must be imported) contains popup notifications that can be used, namely *qtutil.critical*, *qtutil.warning* or *qtutil.info*. The util directory contains functions and classes that are often used or inherited across many plugins (See sect. 3.3). *file_io* contains functions that deal with loading and saving to file, with loading and saving functions that check type, handle memory issues, providing a notification as a file is being saved or written and overwriting protocols when saving a file with the same name. To load a path to use in a plugin, simply use this module. *Project_functions* which is imported as *pfs* contains functions that typically see recurring use by plugins throughout the whole project. These include *save_project* which saves the current state of the project to a .JSON file that organizes all the data saved to file. This alters the .JSON file. *refresh_list*, also in *project_functions*, updates the given UI list to match the altered .JSON. *save_project* saves

the data and *refresh_list* refreshes the UI immediately to reflect the change. It is thus unsurprising that in most plugins a call to *refresh_list* follows somewhere soon after a call to *save_project*.

The code loops through a tuple of the selected items in each list (paths and integers), loads the image stack at the path and adds the integer to it and then saves the result again. Note that *save_project* takes the old *video_path*, the project, the altered image stack and then two strings. The first is the manipulations string and is used to create a new path by taking the old and tacking on the manipulation string to it so that it is clear what processing steps a file has undergone. We have here made 'custom-addition' the name of the process undergone in our plugin. Note also that this function takes a file-type string 'video.' This is used to tag data. If you open the "Manage Data" button in MBE you'll see a 'type' column which takes its output from what you choose as an input for this last parameter in *save_project*. Leave this as 'video' for now.

21. Run MBE and use the import plugin to load in 3 image stacks. Load in your .txt. From your *custom_addition* plugin. Select all the additions and stacks and press 'Perform Addition.' You should see 3 new image stacks with "custom-addition" tacked to their name. These have been added to the list as in Fig. C.4.

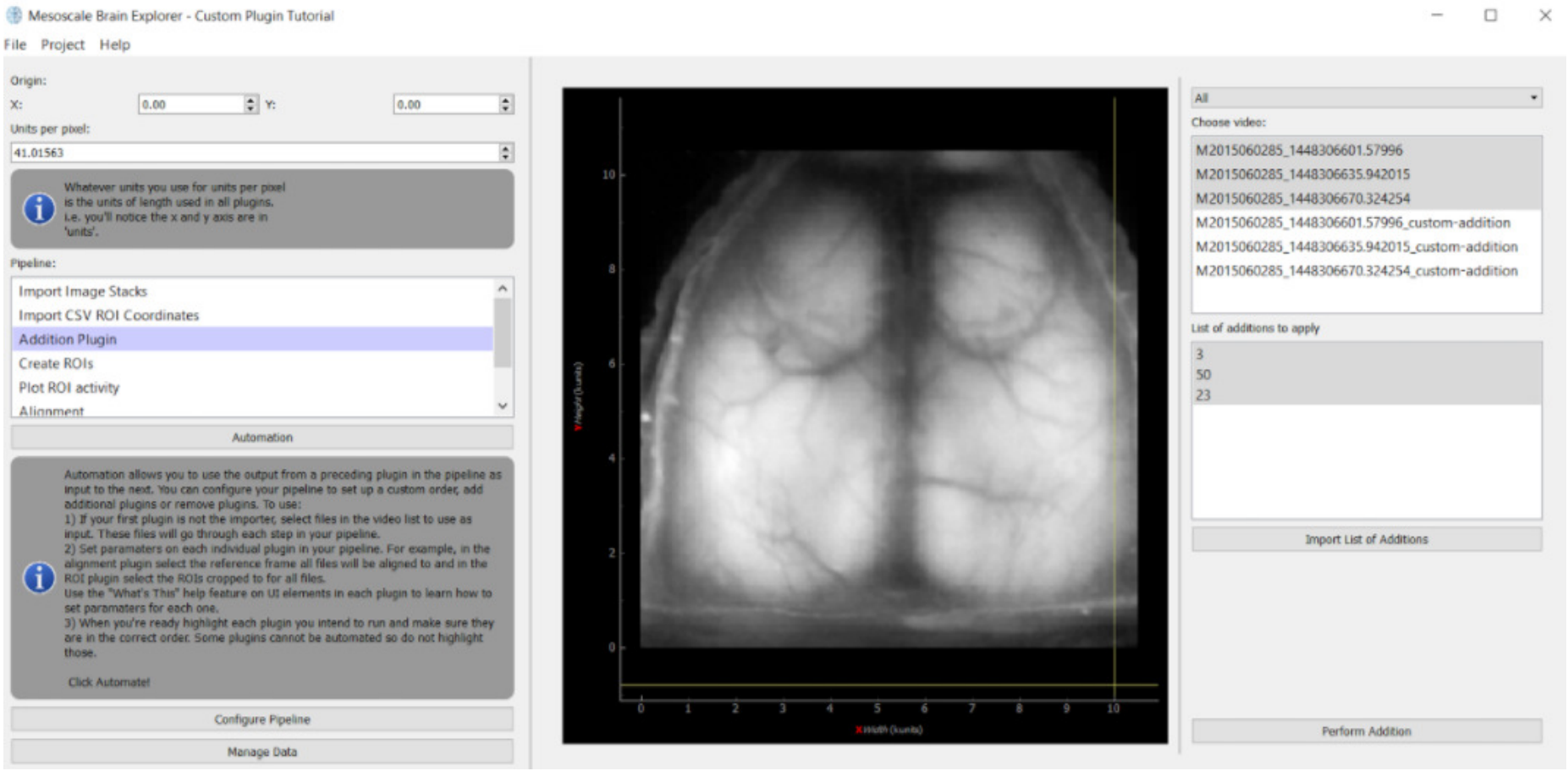


Figure C.4: Addition plugin with three image stacks selected and the three imported numbers (additions to apply) selected. "Perform Addition" has been applied and as such three new image stacks have appeared in the ImageStackListView.

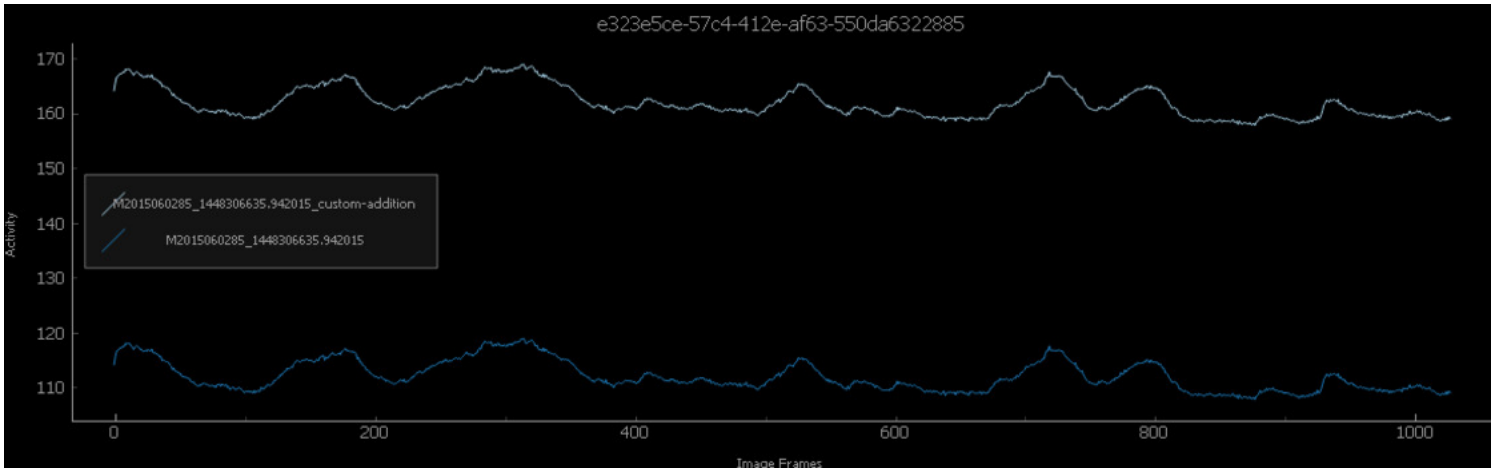


Figure C.5: Activity plot of a randomly selected ROI for mouse #285 before and after the value 50 has beed added to it.

You can create a ROI and plot the activity of the same ROI (see Haupt et al. (2017)) for an image stack before and after the operation to verify that all values in the image stacks with addition have been increased by the expected amount. Fig. C.5 is the output of the second image stack brain activity across an arbitrarily created ROI over brain matter and the same image stack's activity at the same ROI after addition. The activity dynamics has not been changed, however all values have been increased by 50, as expected. Note that even though the values here are clearly different, this is not reflected (as of 0.7.20) when you view these two image stacks in the video player. This is because the video player scales all values that are within each image stack. This makes comparisons between image stacks using the video player unfeasible as each loaded image stack will have its own upper and lower bounds on its scale.

22. Each image stack has a different value that is added to it. This complicates matters as a plugin is generally assumed to encapsulate a single processing step where that single processing step is applied identically to each image stack selected. The output names for each stack merely has 'custom-addition' tacked on and do not allow you to differentiate the image stack that had the value 3 added versus the one that had the value 50 added. To remedy this problem it would be better to change the manipulation to include not only the name of the process, but also the addition being applied for each image stack. Go back to the i*nsert functionality here* todo and modify the call to *save_project* such that it is as follows instead:

```
path = pfs.save_project(video_path, self.project, frames,
    'custom-addition-'+add_num_item.text(), 'video')
```

Rerun MBE and perform the addition again on the same addition again on the same three image stacks (it is normal for now that you have to upload the .txt/.csv each time). The addition applied to each image stack is now plainly visible, allowing you to see exactly what value was added to which

image stack (Fig. C.6):

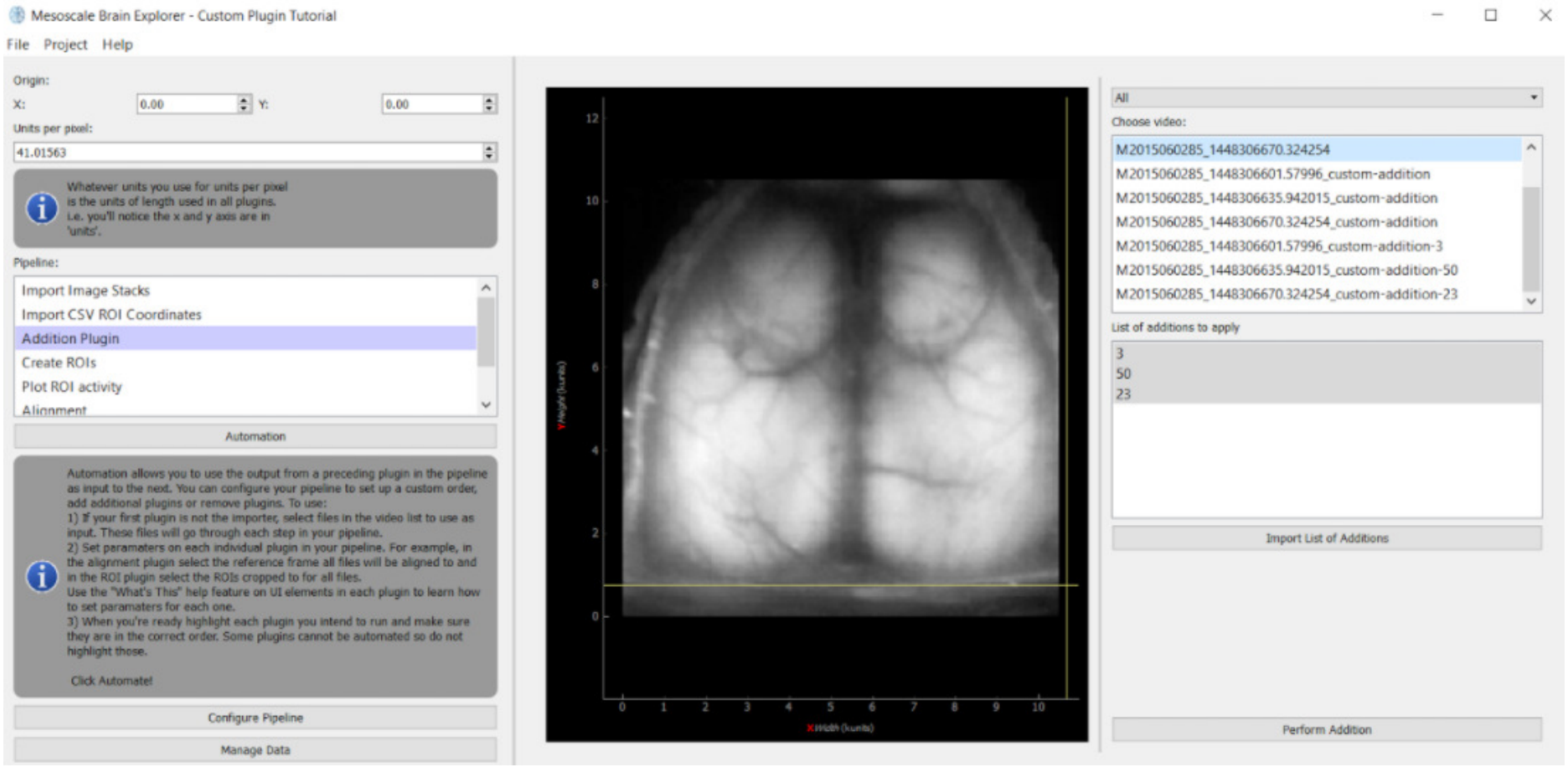


Figure C.6: Addition plugin where the manipulation string "custom-addition" now includes the value that was added to each image stack.

23. The order that files and additions are selected determines which addition is paired with which image stack. So, let us say you have image stacks A, B, C and additions x,y,z both listed in order. If you select B, A and then C and then additions x,y,z then the result will be B+x, A+y, C+z. This is a problem because in the interface there is nothing indicating the order that a user selected items. Despite the user clicking B, then A, then C, the image stacks are still listed A,B,C. One solution would be to create a new list that populates the selected items in the order that they are clicked. Let's implement another two lists to keep track of user selections. Go back to the *Define global attributes and UI components here* todo and define the following two new lists:

```
self.selected_videos_list = QListWidget()
self.selected_add_list = QListWidget()
```

24. Go to the *setup UI component layout and properties* here todo and replace the code for *setup_ui* with the following:

```
def setup_ui(self):
 super().setup_ui()
 # todo: setup UI component layout and properties here
 hbox = QHBoxLayout()
 hbox.addWidget(QLabel("Select Additions"))
 hbox.addWidget(QLabel('Selected Stacks'))
 hbox.addWidget(QLabel('Selected Additions'))
 self.vbox.addLayout(hbox)
 hbox = QHBoxLayout()
 hbox.addWidget(self.add_list)
 hbox.addWidget(self.selected_videos_list)
 hbox.addWidget(self.selected_add_list)
 self.vbox.addLayout(hbox)
 self.add_list.setSelectionMode(QAbstractItemView.ExtendedSelection)
 self.vbox.addWidget(self.import_additions_btn)
 self.vbox.addStretch()
 self.vbox.addWidget(self.add_btn)
```

Note here that we've added a horizontal layout and called it *hbox*. We can add widgets to *hbox* and then finally add the *hbox* to our vertical layout *vbox* which is a defined attribute in *WidgetDefault*, which is the superclass of *Widget*. The above code thus adds 3 labels horizontally, then adds these 3 labels vertically. In the next vertical layer we add 3 lists horizontally:

25. Run MBE to see the result which should have the same layout as Fig. C.7).

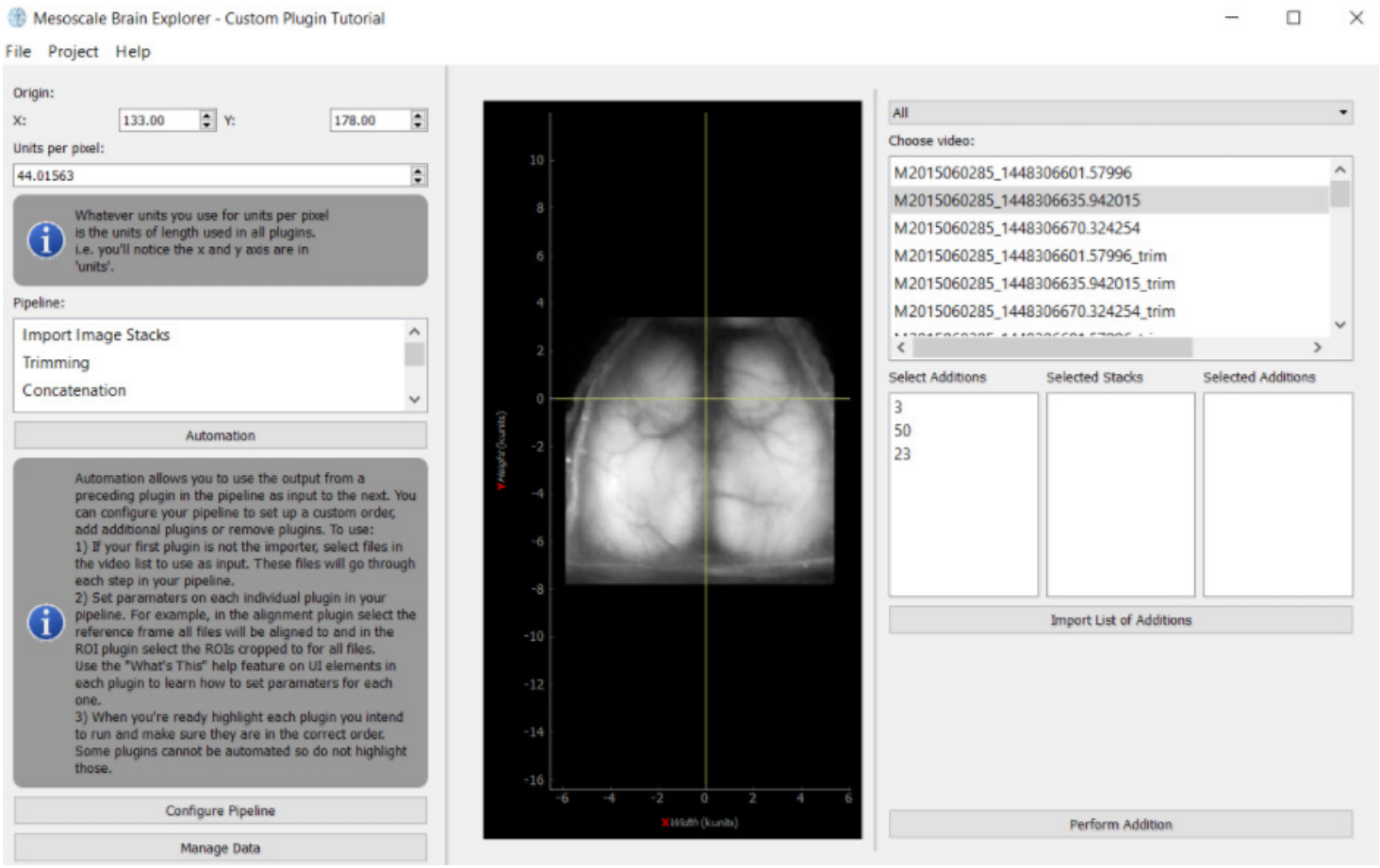


Figure C.7: Custom addition plugin with new three-list UI layout.

26. Go to the *Setup signals (i.e. what ui components do) here* todo and add the following two lines to setup_signals:

```
self.add_list.selectionModel().selectionChanged.connect(
self.add_selection_to_selected_add_list)
self.video_list.selectionModel().selectionChanged.connect(
self.add_selection_to_selected_video_list)
```

Each *QListWidget* has a *QItemSelectionModel* class that keeps track of a view's selected items. We use this to connect a selection change event to a particular function.

27. We need to implement the two functions (*add_selection_to_selected_add_list* and *add_selection_to_selected_video_list*) we've just connected a selection change in the two lists to. Add the following two functions below the *setup_param_signals* function:

```
def add_selection_to_selected_add_list(self, selected, deselected):
  self.selected_add_list.clear()
  self.selected_add_list.addItems([index.data() for index in
     self.add_list.selectedIndexes()])

def add_selection_to_selected_video_list(self, selected, deselected):
  self.selected_videos_list.clear()
  self.selected_videos_list.addItems([index.data() for index in
     self.video_list.selectedIndexes()])
```

When each function is entered, the corresponding "selected_list" is cleared of previously selected items and filled with with all current selected items

28. Run MBE, import the .txt and select 23 and then the number 3. It should appear in this order in the list on the right C.8. Next select two image stack and apply the additions and you'll see that the first image stack you selected is added to the middle list and has the 23 added to while the second has the 3 added as in Fig. C.8.

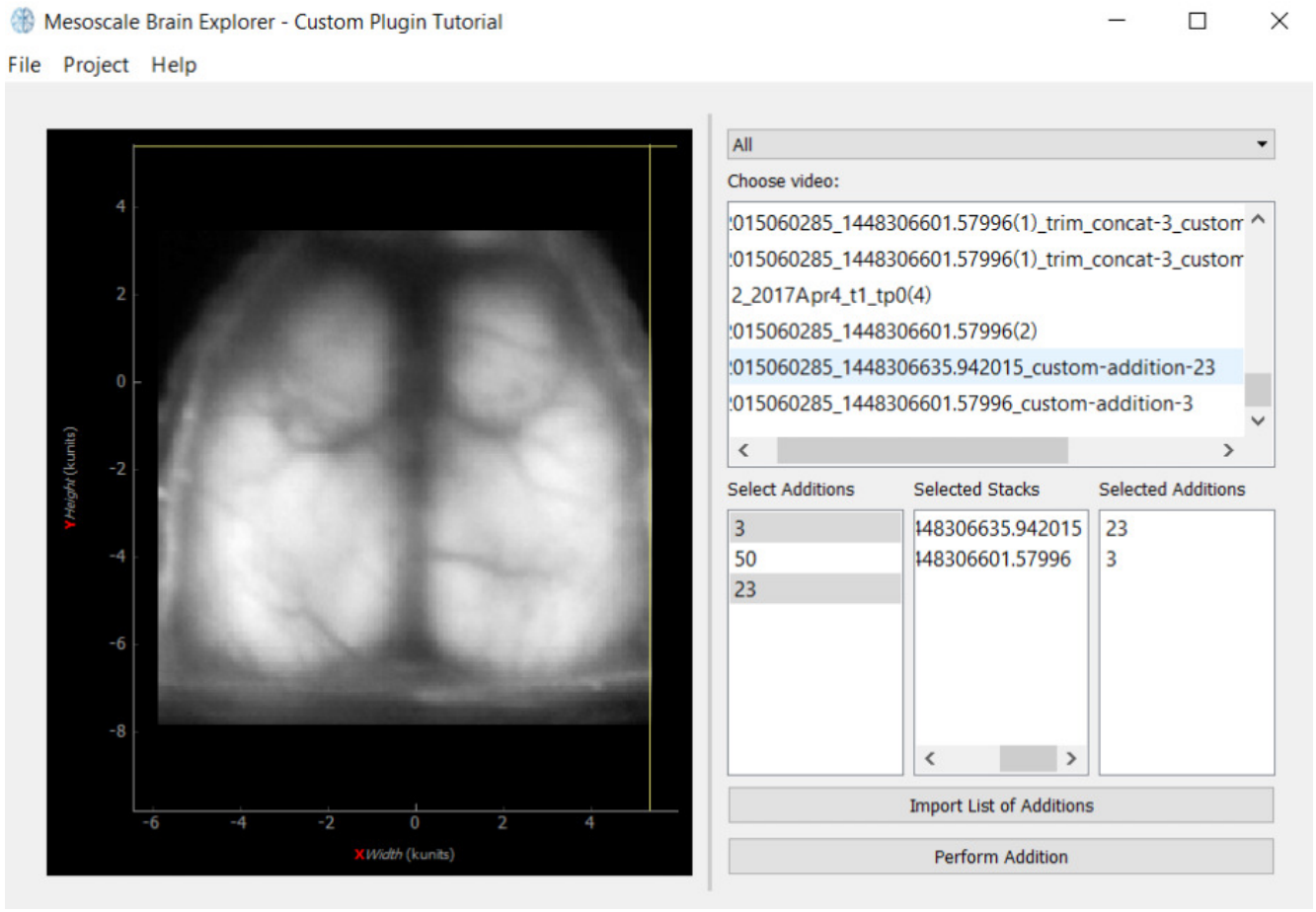


Figure C.8: Custom addition plugin with new three-list layout functional. Values are selected in the "Select Additions" list and the order selected determines where they are added in the "Selected Additions" list. The "Selected Stacks" list has one stack paired with a value in the "Selected Additions" list.

29. How to operate a plugin may be confusing to users. You can add help messages for users. Go to the *setup custom help messages to aid the user, each tied to one of your UI components* todo. Add the following line:

```
self.selected_add_list.setWhatsThis("Additions in this list will be
    applied to the matching stack in the center list")
```

Now when you run MBE and press help then "What's This?" Or Shift+F1 and click on the *selected_add_list* UI component you'll be greeted by your message as in Fig. C.9. Congratulations, you have the first workings of your own custom plugin!

C.3 Part 2 - Parameter Persistence

You may have noticed that after importing the list of additions that the values do not persistently stay in the *add_list*. Instead you are faced with the arduous task of importing it each time you want to use the plugin. Moreover, from other plugins you should have noticed that you can set the parameters of most UI components (i.e set values in the plugin user interface components section of the plugin pane (Fig. 3.2)) and then navigate away and navigate back and your parameter settings will be saved. To make parameters persistent for Addition Plugin we need two pieces of information: the location of the text file that contains the the addition numbers and the indexes of the list that the user has selected. Both need to be made persistent. Parameter persistence is also a key step to allow the Addition Plugin to be automated in a pipeline which is covered in C.4.

1. Go to the *Define labels used as a key to save parameters to file here* todo and change the Labels class to read as follows:

```
class Labels(WidgetDefault.Labels):
 add_list_path_label = 'add_list_path'
 add_list_index_label = 'Select Additions'
```

This class contains keys that will be used to retrieve the items in the Python dictionary *params* that contain the plugin parameter values stored to file. Here we use unique string identifiers. In the second line note how we use the label for *add_list* as the key since it is unique to the plugin and is already used in the interface to identify the UI component.

2. Go to *setup_ui* and change the following line

```
hbox.addWidget(QLabel('Select Additions'))
```

to

```
hbox.addWidget(QLabel(self.Labels.add_list_index_label))
```

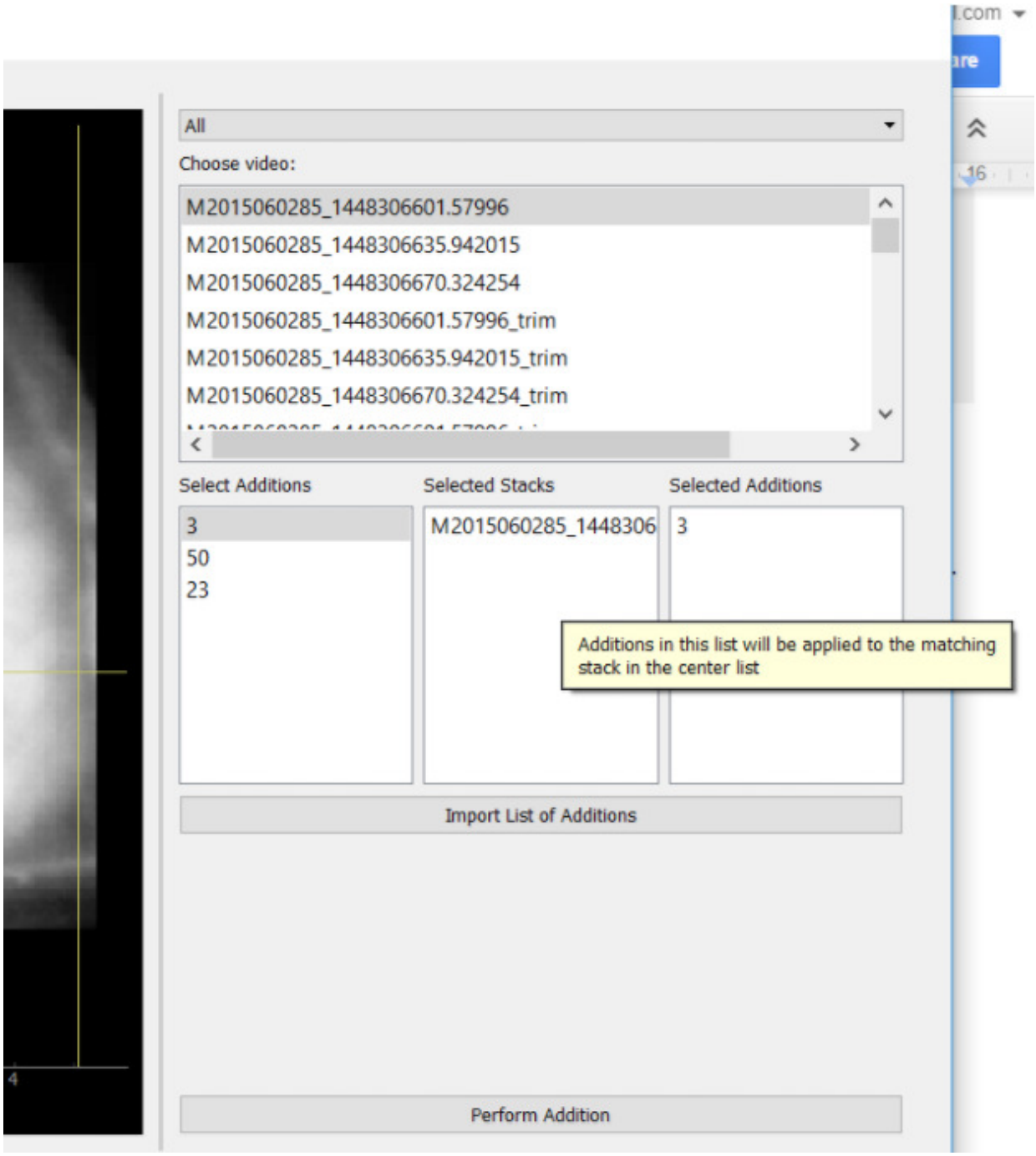


Figure C.9: Zoomed in screenshot of the custom addition plugin with a "What's this" contextual help message implemented.

It is generally better to keep strings in one place, in this case the Label class. Then, if the string needs to change it only needs to be changed in one location.

3. Next we need to define default values. These are the initial values that a plugin's UI components take before any user interaction. Go to the *Define default values for this plugin and its UI components here* todo. Update the Defaults class such that it looks like this:

```
class Defaults(WidgetDefault.Defaults):
 add_list_path_default = QSettings().value('last_load_text_path')
 add_list_index_default = [0]
```

The first default is simply the path to the last loaded location where the key 'last_load_text_path' was used to save QSettings. This key is only used throughout MBE to store the location of text files so this default points to the last loaded text file location. The index default is a list since multiple values can be selected. The default selection is to only have the first index selected, therefore the list contains a single zero.

4. Go to the *setup plugin parameters (e.g. UI components starting values) initial values here* todo. Delete any code inside of the if statement and add these two lines instead:

```
self.update_plugin_params(self.Labels.add_list_index_label,
    self.Defaults.add_list_index_default)
self.update_plugin_params(self.Labels.add_list_path_label,
    self.Defaults.add_list_path_default)
```

The code inside of the if statement is what is run the first time a plugin is set up. At this point the default values have not been saved to the *params* variable yet so it will have a length of 1. The *update_plugin_params* function is used and given a key and value pair which is saved to a .JSON file that organizes all files in the project.

5. Go to the *setup where plugin parameters get their values from* todo which is in the same function as step 4, just outside of the if statement. Delete any code outside the if statement and add these lines instead:

```
self.import_additions(self.params[self.Labels.add_list_path_label])
add_list_indices = self.params[self.Labels.add_list_index_label]
pfs.refresh_list(self.project, self.add_list, add_list_indices)
```

These are the lines that actually set the plugin parameter values in the UI. The lines in the if statement merely calls *update_plugin_params* to save them to file. We need to update the values in the add list and populate it with values from a text file if a text file is already saved in the *params* variable. So, here we manually call the *import_additions* function and give it a parameter that is a call to *params* with the label for the *add_list_path*. If a file location has already been saved then we

will call *import_additions* to populate the list. Likewise we call *refresh_list* with previously selected indexes saved in *params* afterwards to make sure that *add_list* still has the same indexes selected when a user navigates to another plugin and back - or even if they exit MBE and restart it.

6. In step 5 we are calling *import_additions* with an argument despite it taking no arguments. Remedy this by replacing *import_additions* with the following:

```
def import_additions(self, text_file_path=None):
  if not text_file_path:
     text_file_path = QFileDialog.getOpenFileName(
        self, 'Load images', QSettings().value('last_load_text_path'),
        'Video files (*.csv *.txt)')
     if not text_file_path:
        return
     QSettings().setValue('last_load_text_path',
        os.path.dirname(text_file_path))
     copyfile(text_file_path, os.path.join(self.project.path,
        os.path.basename(text_file_path)))
     text_file_path = os.path.join(self.project.path, text_file_path)
     self.update_plugin_params(self.Labels.add_list_path_label,
        text_file_path)

  if text_file_path.endswith('csv') or text_file_path.endswith('txt'):
     add_list = [] # numpy way: np.empty(shape=(1, ))
     with open(text_file_path, 'rt', encoding='ascii') as csvfile:
       add_list_it = csv.reader(csvfile)
       for row in add_list_it:
         add_list = add_list + row
     self.add_list.addItems(add_list)
```

This should be mostly identical to the code you already have, except that two new if statements have been added as well as a call to *update_plugin_params* that stores the selected file's path. The first if statement one checks whether *text_file_path* has a value. If it doesn't then the user is clicking the button and so we trigger the *QFileDialog* and let the user select the location of the file. If *text_file_path* however has a value then the function if being called with a value stored in *params*. In this case we don't need to request a path from the user since we already have the the path to the file. We just have to load itand populate *add_list* which is what the latter half of this function accomplishes. We additionally now need to make sure that *text_file_path* is a .csv or .txt.

7. When the user interacts with the *add_list* UI nothing determines that the new selected indexes should be saved. Try running MBE for yourself. Import the text file. Now navigate away to another plugin and navigate back to the Addition Plugin. Success! You'll notice the values for the *add_list* have been automatically re-populated. I.e. its values are now persistent. However, select all the values in

your *add_list*. Now navigate to another plugin and then navigate back to Addition Plugin. You'll notice that the selection has reverted back to the default of only the first index being selected. We still need to make selected indexes stay persistently selected. Go to the *setup how UI component interaction triggers parameter storing* todo. Add the following line of code that connects user changing the selection of *add_list* with a new function called *update_add_list_index*:

```
self.add_list.selectionModel().selectionChanged.connect(
self.update_add_list_index)
```

8. Now implement *update_add_list_index*. Add this new function to the *Widget* class:

```
def update_add_list_index(self, selected, deselected):
  val = [v.row() for v in self.add_list.selectedIndexes()]
  self.update_plugin_params(self.Labels.add_list_index_label, val)
```

   All selected indexes are collected and *update_plugin_params* is called to save these indexes to *params* using the label for indexes. So when *setup_params* is called again, this line: *add_list_indices = self.params[self.Labels.add_list_index_label]* will retrieve those stored values.

9. Rerun MBE and you'll notice that the Addition Plugin's parameters are fully persistent!

C.4 Part 3 - Automation

Now that relevant plugin parameters are persistently kept upon user interaction we can rather easily automate data processing. What this will enable is to add the Addition Plugin anywhere in a custom pipeline and with one click go through all steps, including the Addition Plugin step, in that pipeline.

1. The *MyPlugin* class inherits from *PluginDefault*. Go to *PluginDefault* in *src/plugins/util/plugin.py* and you'll see the following

```
def check_ready_for_automation(self, expected_input_number):
  return False

def automation_error_message(self):
  return "Plugin " + self.name + " is not suitable for automation."
```

   The default behaviour for a plugin therefore is to return false when checked if it is ready for automation and to return an error message that it is not suitable for automation. We need to override both of these

2. Go to the *override PluginDefault functions here to define custom behaviour* todo. Add the following two functions:

```
def check_ready_for_automation(self, expected_input_number):
  if len(self.widget.add_list.selectedIndexes()) == expected_input_number:
     return True
  else:
     return False

def automation_error_message(self):
  return "Plugin " + self.name + " cannot work if the number of selected
     additions do not equal the number of selected image stacks"
```

When the plugin is checked for whether it is ready for automation it checks whether the number of inputs the plugin is expected to receive matches the number of indexes selected in *add_list*. If not then automation should not proceed. We don't want to proceed through a custom pipeline where the Addition Plugin is but one intermediate step in this pipeline only to find after we get to the Addition Plugin that it failed. The *check_ready_for_automation* function therefore checks each plugin selected for automation before automation is allowed to commence.

3. That is all there is to it. Try rerunning the application on a new project (delete the old one). Click 'Configure Pipeline' and pick a custom pipeline. For example as in Fig. C.10. Now import the text file and select the three additions as in Fig. C.11.

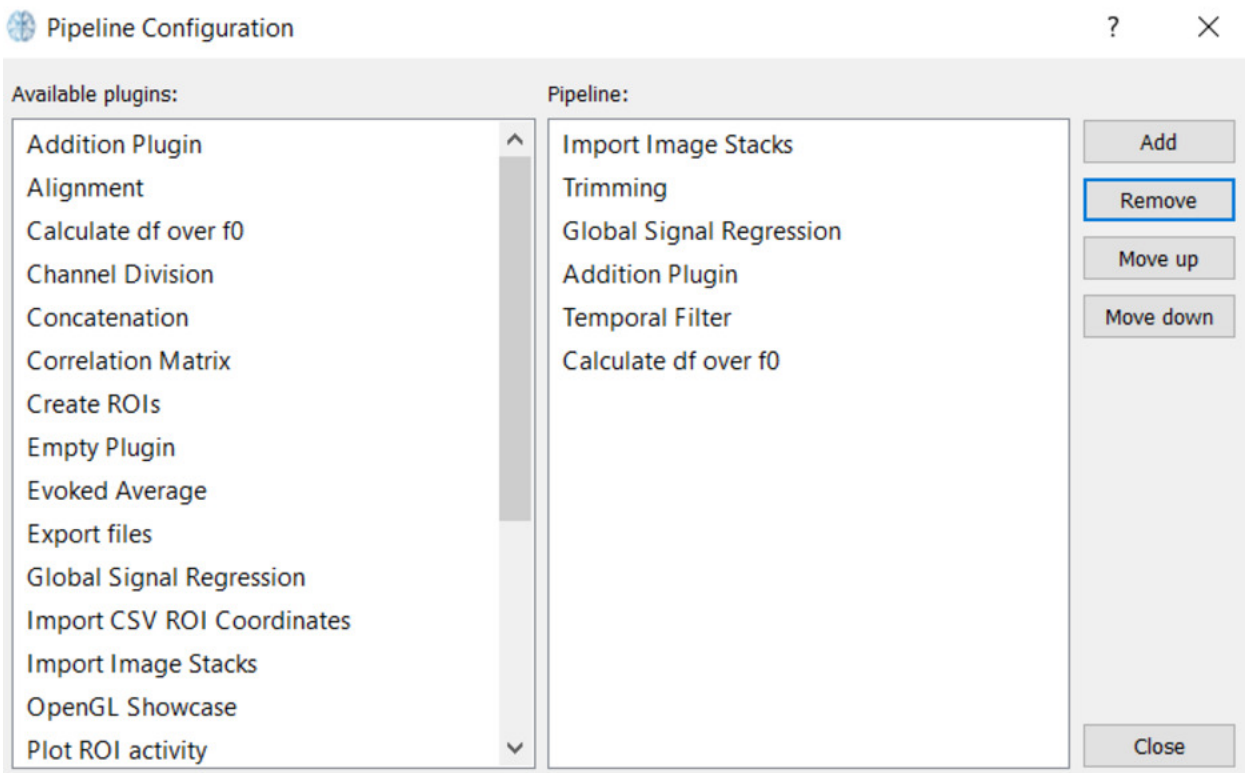


Figure C.10: The Pipeline Configuration window with a pipeline example where all added plugins can be automated.

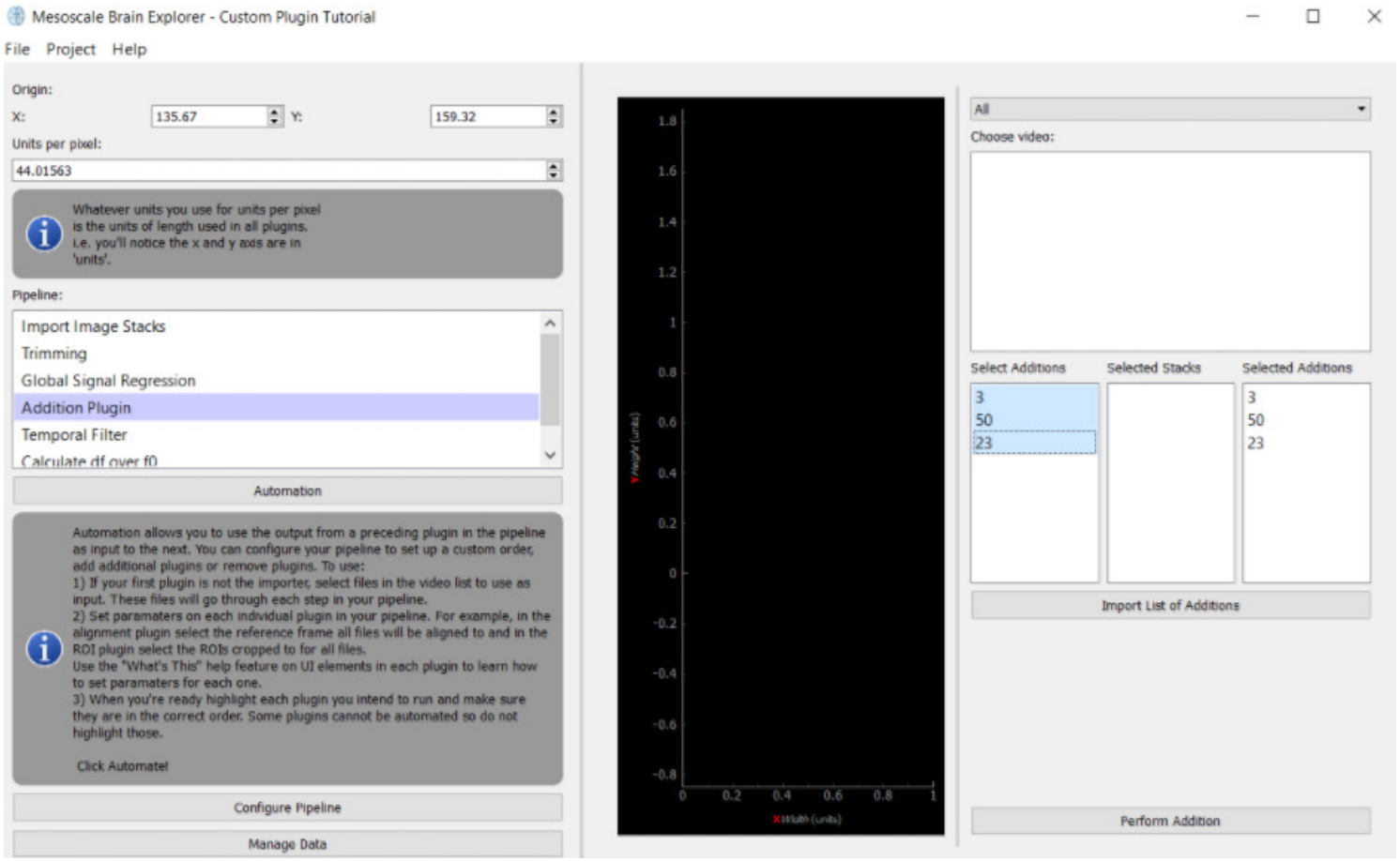


Figure C.11: Addition Plugin selected and three imported numbers selected.

Select the plugins that will used (and select the appropriate number of image stacks if not using the Import Image Stack plugin) as in Fig. C.12.

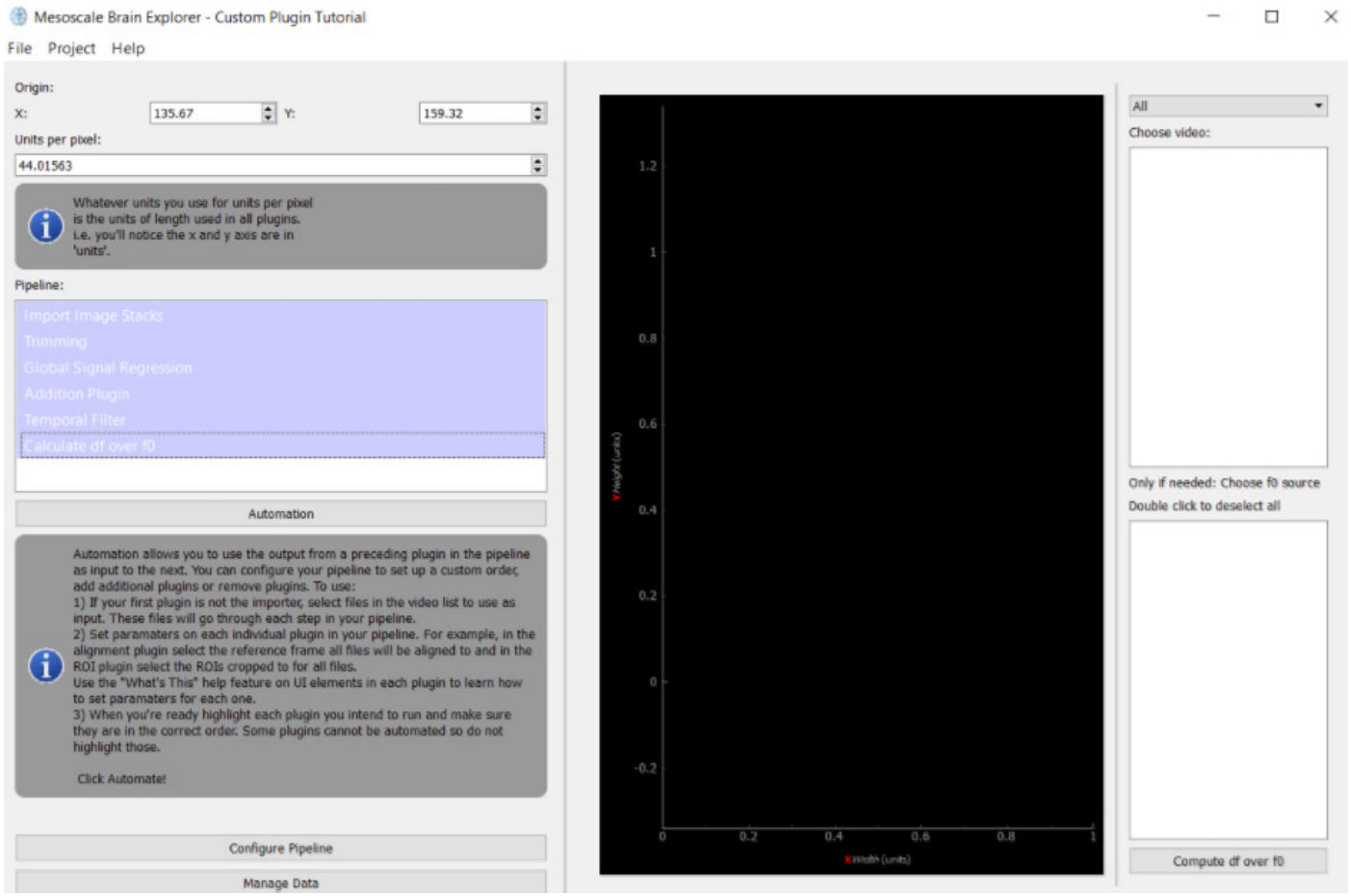


Figure C.12: All plugins in the pipeline example selected for automation set up. Paramaters for each one is set. The pipeline is navigated (from plugin to plugin) without disrupting the selection using the right mouse button.

Press automate. After processing is complete you can navigate to near any plugin to see all the files that were generated through the course of the processing pipeline. The result of our automated pipeline can be seen in Fig. C.13).

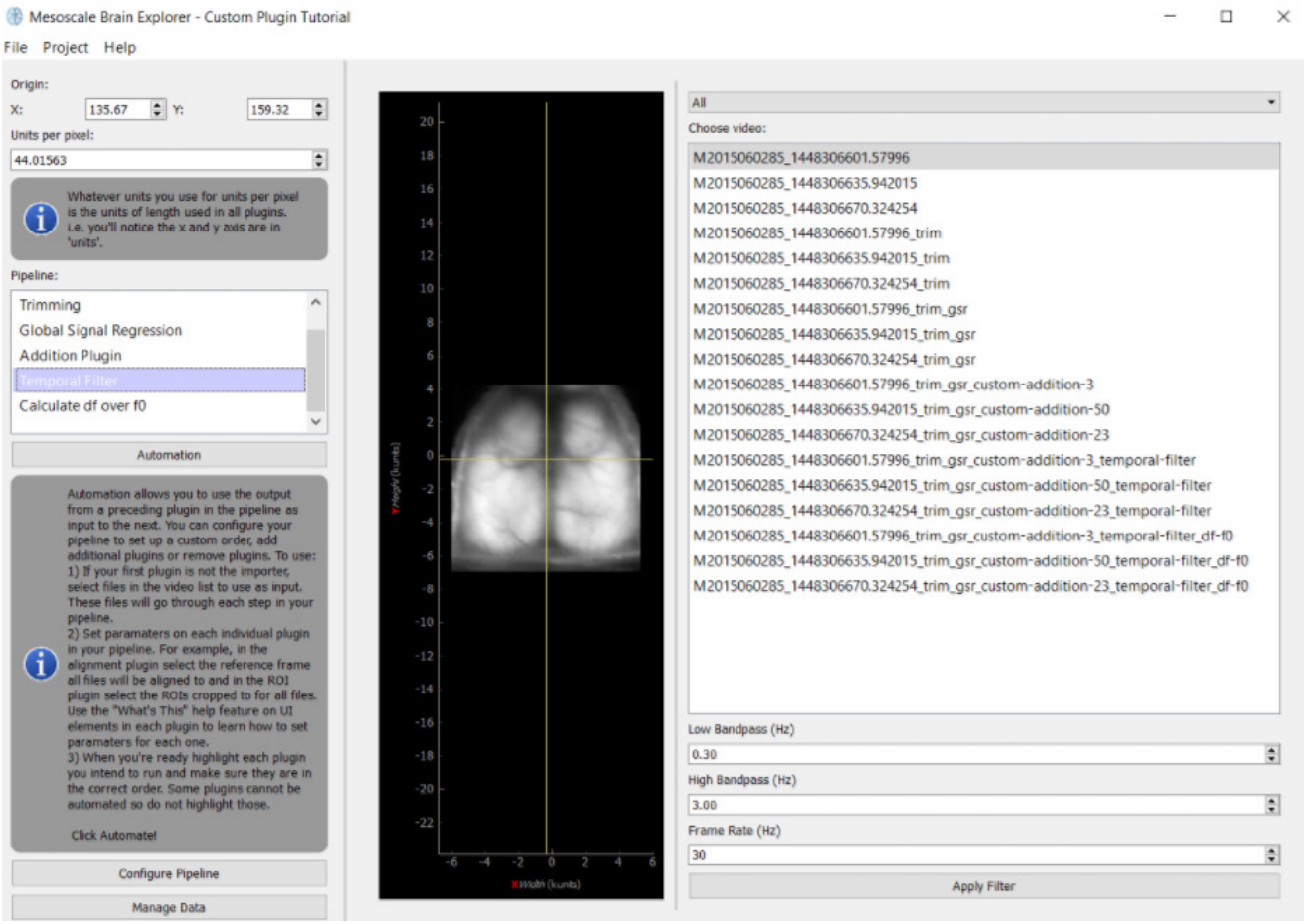


Figure C.13: Final result of automating the example pipeline where three image stacks were each trimmed, then GSR applied, then a custom addition value added, then a temporal filter and finally $\Delta F/F_0$.

C.5 Building an Executable

Running the terminal in the same directory as pipegui.py the following command can be used to package the code into an executable, provided that Pyinstaller is installed and its executable "pyinstaller.exe" is at the location called:

```
C:\Python35\Scripts\pyinstaller.exe --additional-hooks-dir=.
--clean --win-private-assemblies --onedir pipegui.spec
```

A folder called "dist" will be created. This folder will contain a single folder "pipegui." The pipegui.exe file in this folder will not work yet as it cannot find files it expects one directory above it. First, the LICENSE and VERSION files along with the templates directory must by copy-pasted from their location above the src directory to the same level as the pipegui directory (one level above your new .exe). After this step your executable should run.