Physical and computational methods of investigating the relationship between stress, cognition, and behavior in the context of the KOR-dynorphin system

Mackenzie M Andrews

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Bioengineering

University of Washington

2019

Committee:

Andre Berdt

Charles Chavkin

Program Authorized to Offer Degree:

Bioengineering

University of Washington

Abstract

Physical and computational methods of investigating the relationship between stress, cognition, and behavior in the context of the KOR-dynorphin system

Mackenzie M Andrews

Chair of the Supervisory Committee:

Andre Berndt

Department of Bioengineering

Motivation: Chronic stress is an aversive experience that can have maladaptive effects on an individual's cognition and behavior, ultimately leading to increased risk of developing conditions such as anxiety, depression, or amplifying symptoms of drug use disorder. Stress-induced release of dynorphin activates the kappa opioid receptor (KOR), causing short- and long-term disturbances at the behavioral, circuit, and cellular levels.

Contents: This thesis discusses experimental and computational methods to probe the KOR-dynorphin system at the behavioral, circuit, and cellular levels are described. At the behavioral level, the differential reinforcement of low response rate and delayed alternation tasks are used to investigate KOR effects on aspects of motivated behavior. At the circuit level, *in vivo* fiber photometry is used to illuminate important aspects of reward signaling following stress. Finally, at the circuit level, real-time cellular microscopy is used to investigate the cellular pathways activated via KOR, providing potential therapeutic targets for stress-induced pathophysiologies.

Results: The behavioral assays demonstrated that KOR activation causes cognitive disruptions that affect both inhibitory control as well as working memory-dependent decision making. The fiber photometry results demonstrated that stress causes a KOR dependent increase in dopamine neuron activity in a major reward center, following the stress exposure. Finally, the real-time cellular measurements revealed a new branch of G-protein coupled receptor signaling that results in production of reactive oxygen species which leads to permanent KOR inactivation.

Dedication

To my mother, for your never-ending support and ever-shining light,
even in the darkest of rooms.

To my father, for sparking my scientific curiosities and igniting the
passion driving my motivation.

To Mrs. Stuckey and Mr. Bassett for equipping a generation of youth with the
excitement and foundation to solve this world's greatest mysteries.

To Dr. Antony Abraham for exemplifying the qualities of an invaluable mentor through your
enthusiasm, patience, kindness, and friendship.

Bad times have a scientific value. These are occasions a good learner would not miss.
-Ralph Waldo Emerson

Santa Rosa Fairies

The needle is a heaven
Piercing a damp cardboard reality
Penetrating the tulip heart of a rebel
A hill of sprouting fairies
Singing lullabies to the Devil

The needle is a hell
A volcano spewing black tar
He knows the feathers let him fly
Escape the clawing disease of thought
As china white snow turns dry

The sky bursts to flame
As hands reach for the number 8
The charcoal rain has begun
Young sees the needle
And the damage done

Tears fall behind rolling eyes
Tunneling under pale cheek skin
Raining on a cracked tongue
Who spits “we all are
like a setting sun”

"Don't worry though my dear"
Whisper brown eyes turning grey
"Take this rag and polish up the spoon
And I will be singing
With the fairies soon"

Santa Rosa Sunset

The flame cauterizes the pain
        the guilt, the shame
Setting fire to the spoon
        a ladle for the joyous stew
Melting; tar from vows,
        cotton from clouds
A heavenly vacancy
        drawing up, up impatiently
Pierce the freckled leather
        pushing down, down into the nether

Tomorrow’s effluent flows through veins
        warming, soft, smile again
He sings the song from wrinkled jowls
        “I love you baby” the dry voice howls
A pelvic swing starts the dance
        a 12-step tango, fairies prance
The bunny hop turns to Easter eggs
        all about run little legs
A gentle embrace from his giggling girl
        assure him this is a better world

But fairies turn to skunks
        children become drunks
Rosy cheeks and illuminating smiles
        now missing teeth and yellow bile
Chew through tongues and cracking lips
        “some more, more” desperation spits
Men in black load up cars
        phobic hands cling to metal bars
Weary knees drop to the floor
        wheel flanges screech, locking doors

Blind to what has fallen away
        The sun sets on another day

Metamorphosis

In the room of a poisoned man, trapped
within lavender walls
dusty stuffed animals shoved to a corner
under sparkling bottles reeking of wasted spirits. Below,
a shell of sun hardened leather and soggy lips
grey with empty promises, shivers

within a cocoon of butterfly blankets--
drowning in a toxic rain, he never
grew his wings

Tracks of purplish blackish veins, thirsty
tunneling under his skin, searching
for a way out of this pulsing carcass

Two helpless grey eyes roll
towards me within blackened holes--
I watch my reflection, dancing
on those glossy frozen lakes

Four smiles emerge--one
breaking through cracking skin
broken teeth, yellowed tongue
spitting and choking, two and three
on the little girls gliding
across iris fields, staring
back at me, four
wings uncurling from the corners of my mouth
soft, pink--I am

the butterfly.

2019

Physical and computational methods of investigating the relationship between stress, cognition, and behavior in the context of the KOR-dynorphin system

MASTER OF SCIENCE IN BIOENGINEERING
UNIVERSITY OF WASHINGTON

MASTER OF SCIENCE IN BIOENGINEERING, UNIVERSITY OF WASHINGTON

TABLE OF CONTENTS

ABSTRACT

Motivation: Chronic stress is an aversive experience that can have maladaptive effects on an individual's cognition and behavior, ultimately leading to increased risk of developing conditions such as anxiety, depression, or amplifying symptoms of drug use disorder. Stress-induced release of dynorphin activates the kappa opioid receptor (KOR), causing short- and long-term disturbances at the behavioral, circuit, and cellular levels.

Contents: This thesis discusses experimental and computational methods to probe the KOR-dynorphin system at the behavioral, circuit, and cellular levels are described. At the behavioral level, the differential reinforcement of low response rate and delayed alternation tasks are used to investigate KOR effects on aspects of motivated behavior. At the circuit level, *in vivo* fiber photometry is used to illuminate important aspects of reward signaling following stress. Finally, at the circuit level, real-time cellular microscopy is used to investigate the cellular pathways activated via KOR, providing potential therapeutic targets for stress-induced pathophysiologies.

Results: The behavioral assays demonstrated that KOR activation causes cognitive disruptions that affect both inhibitory control as well as working memory-dependent decision making. The fiber photometry results demonstrated that stress causes a KOR dependent increase in dopamine neuron activity in a major reward center, following the stress exposure. Finally, the real-time cellular measurements revealed a new branch of G-protein coupled receptor signaling that results in production of reactive oxygen species which leads to permanent KOR inactivation.

CHAPTER 1 – INTRODUCTION

OVERVIEW

Chapter 1 introduces the contextually significant background related to the relationships between stress, cognition, and behavior. This information details the behavioral, cognitive, and physiological effects of acute stress as well as the clinically relevant effects of severe or chronic stress on mood disorders and drug abuse and relapse. Stemming from this context, the importance of the kappa opioid receptor (KOR) and its endogenous activator, dynorphin, on the aforementioned stress effects is described. A broad understanding of the behavioral, circuit, and cellular levels of KOR action is developed with an emphasis on the complexity of the KOR-dynorphin system at each level of investigation.

The complexity of KOR action introduces the motivational context surrounding data driven research. The arms race between investigative tools and data analysis methods is described with the conclusion being that analysis methods are struggling to keep up with the technological advancements of investigative tools. This analysis bottleneck motivates the need for computational tools that can handle, process, analyze, and communicate large, multi-variate, heterogenous data sets.

With experimental context critical for appropriate computational development, the experimental approaches used throughout this thesis are introduced. Investigative methods of behavior, circuit dynamics, and cellular pathways are described with an emphasis on the gaps between classical analysis methods and the extensive data available.

Finally, the state-of-understanding prior to the work of this thesis is depicted and key pharmacological agents are defined.

BACKGROUND AND SIGNIFICANCE

STRESS, COGNITION, AND BEHAVIOR

Stress plays a critical role in survival. Stress signaling occurs as a result of a challenge or demand and can scale with the severity of the stressor. Small bursts of stress can help individuals escape a dangerous or difficult situation and are followed by a positive emotion of relief or even excitement. However, persistent, uncontrolled stress is highly aversive and can have negative effects on the individual's motivation and reward circuitry (Gold and Chrousos, 2002; Volkow and Li, 2004; Fox et al., 2007; Koob and Kreek, 2007).

Stress evokes acute behavioral, cognitive, and physiological changes, typically associated with fear responses. These stressors can evoke either defensive or escape behaviors – typically

nominalized as the 'Fight or Flight Response.' This behavioral response is stereotypically coupled with affective changes such as a shift to distress or fear, as well as apathy towards the non-immediate situation (Gold et al., 1998). In addition, the acute response to danger involves physiologic changes such as increased heart rate, redirection of blood flow, and inhibition of neurovegetative functions such as feeding, sleep, growth, and reproduction. Following a stressful experience, individuals often have an affective shift towards reward or relief and a reinstatement, or in some cases, over-instatement of neurovegetative functions (e.g. increased eating or sleep) (Lazarus, 1999).

Acutely, these changes in behavior and physiology are temporary and beneficial for rapid survival decisions. However, severe or chronic stress is known to cause cognitive disruptions and increase the risk of mood disorders like depression and anxiety as well as the risk of drug abuse and relapse (Campeau et al., 2011; Gold and Chrousos, 2002; Volkow and Li 2004; Kreek and Koob, 2007; Sinha, 2001). In the case of mood disorders, typical symptoms include generalized anxiety and/or apathy, consistent with the affective changes observed during the stress response. In addition, physiologic symptoms of mood disorders such as disruptions to appetite, sleep, and arousal are also consistent with the physiologic effects of stress.

In the case of drug abuse and relapse, alterations in motivation and reward are associated with an increased risk of use escalation or relapse in drug seeking behaviors (Piazza et al., 1990; Shaham and Stewart, 1994). The affective and physiologic changes associated with both acute and chronic stress have been shown to increase addictive tendencies, however there are a number of theories about the exact mechanisms behind this phenomenon (Tomkins, 1966; Russell and Mehrabian, 1975; Leventhal and Cleary, 1980; Shiffman, 1982; Marlatt and Gordon, 1985; Wills and Shiffman, 1985; Koob and Le Moal, 1997).

THE KOR-DYNORPHIN SYSTEM

The classical stress response described above is driven by the activation of the hypothalamic-pituitary-adrenal (HPA) axis by corticotrophin releasing factor (CRF). The HPA axis controls the stereotypical physiologic effects of stress. However, CRF also has intracerebral targets which result in the cognitive and emotional stress effects. Specifically, CRF induces activation of a class of endogenous opioid peptides, dynorphins. Dynorphins activate the kappa opioid receptor (KOR) to produce the dysphoric effects of stress as well as promote the rewarding effects of drugs of abuse (Land et al., 2008; McLaughlin et al., 2003).

At the behavioral level, animals lacking KOR and animals treated with KOR antagonists show stress-resilience when assessed for anxiety, aversion, and drug reinstatement (Van't Veer and Carlezon Jr., 2013). Additionally, activation of KOR in humans produces aversive and depressive-like emotional states coupled with cognitive disruptions (Pfeiffer et al., 1986). This suggests that KOR plays a critical role in the relationship between stress and mood disorders and drug use disorders. Alternatively, KOR activation has been shown to produce analgesia without the proaddictive effects mu opioid agonists (Shippenberg and Herz, 1986; Pfeiffer at al., 1986).

The effects of KOR on behavior are broad with studies showing clinical potential for both KOR-antagonists as well as KOR-agonists.

At the circuit level, KOR is expressed in a number of brain regions classically associated with motivation, mood, and cognition. Namely, the mesolimbic system is a major driver of affective behavior with KOR playing an important role in this circuit's dynamics. The mesolimbic system originates in the ventral tegmental area (VTA) with projections to the nucleus accumbens (NAc), hippocampus, amygdala, prefrontal cortex (PFC), and bed nucleus of the stria terminalis (BNST) (Swanson, 1982). Inputs from the dorsal raphe nucleus (DRN) to the NAc are also implicated in KOR mediated affective behavior. KOR is highly expressed in the DRN, VTA, NAc, PFC, and BNST and regulates serotonergic, dopaminergic and glutamatergic inputs throughout this system, leading to affective changes and behavioral effects (Figure 1; adapted from Van't Veer and Carlezon WA Jr., 2013).

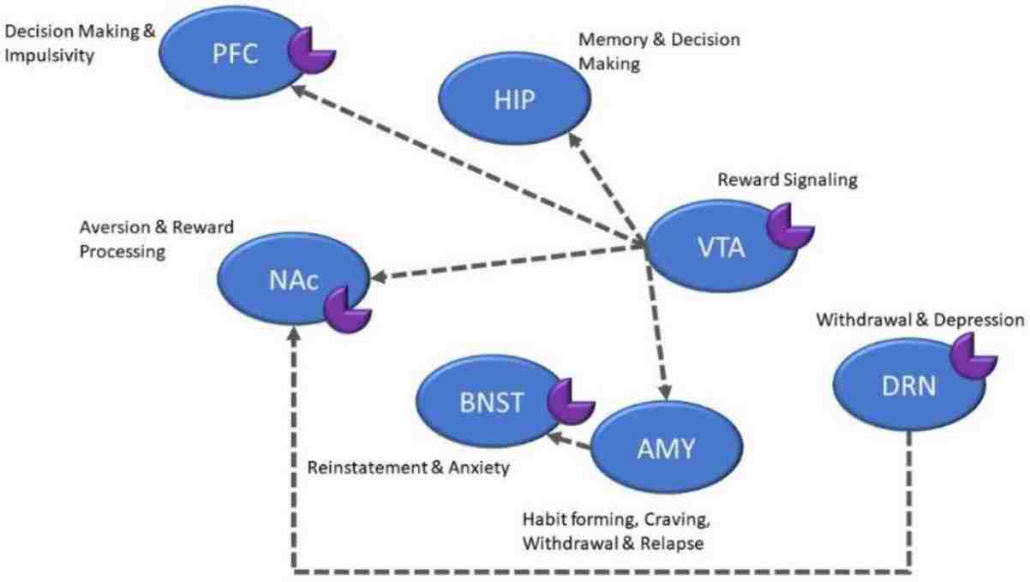

Figure 1: KOR Expression and Behavioral Outputs of the Mesolimbic System

Schematic illustrating the mesocorticolimbic brain areas involved in various aspects of information processing and motivated behaviors. Regions that have been identified to have high expression of KOR are denoted with the 'pac-man' receptor symbol. AMY, amygdala; BNST, bed nucleus of the stria terminalis; DRN, dorsal raphe nucleus; HIP, hippocampus; PFC, prefrontal cortex; NAc, nucleus accumbens; VTA, ventral tegmental area. Adapted from Van't Veer and Carlezon WA Jr., 2013.

At the cellular level, KOR stimulation activates two signaling pathways – the Gβγ-mediated pathway and the GRK/arrestin-mediated p38 MAPK pathway. Studies using Gβγ selective KOR agonists produce analgesia without the dysphoric or cognitive effects associated with unbiased KOR activation (Bruchas and Chavkin, 2010). Better understanding of the cellular mechanisms of KOR activation will provide insight into the circuit dynamics and behavioral effects associated with stress, cognition, and behavior.

MOTIVATION

Investigating the KOR-dynorphin system provides insight that will allow for the development of better clinical therapeutics to treat pain, mood disorders, and addiction. However, the KOR-dynorphin system is integrated into a greater multi-dimensional biological system with highly sensitive and variable outputs for a given series of inputs. As introduced, current investigational strategies approach questions at the behavioral, circuit, and cellular levels. Fortunately, the state-of-the-art investigational tools are producing larger, more complex data sets, allowing researchers to probe more complex questions with greater consistency and resolution of results. However, the technology available to researchers has been advancing so rapidly, that the analysis methods have been struggling to keep up (Mattmann, 2013; Marx, 2013).

Currently, there is a major bottleneck in biological research between data acquisition and data analysis. While other fields such as physics, astronomy, and computer science have been handling large data sets for many decades, biological data sets are significantly more heterogeneous and multivariate, thus requiring a new degree of computational methods (Marx, 2013). To complicate the problem further, biological research is significantly less centralized and standardized than other fields – there are thousands of biology labs developing new experiments and probing at systems in new ways; standardizing the analytic methods for this vastly heterogenous discipline is a daunting yet critical task.

There are four major challenges in computational biology: data handling, data processing, data analysis, and data communication (Alyass et al., 2015). Data handling refers to the movement, storage, and organization of data. As data sets grow in size, storage and movement of that data becomes cumbersome and time-intensive. As the complexity increases, organizing the data in a logical and useful way becomes challenging. Data processing refers to the series of steps that convert 'raw data' into 'usable data.' Often times, biological data are noisy and represent outputs from a range of inputs. Processing this data requires extensive filtering and the accounting for uncontrolled inputs. Data analysis refers to the quantification of the data to answer the posed research question. Analysis requires both computational and statistical methods to appropriately interpret the present information. Finally, data communication begins at computation – to effectively communicate results, data must be displayed in an accessible and visually self-evident way; visual representation of data is a computational challenge in and of itself.

EXPERIMENTAL APPROACH AND NEEDS ASSESSMENT

In order to appropriately handle, process, analyze and communicate data, an understanding of the experimental tools and methods is necessary – a "new breed of researcher equally familiar with science and advanced computing" (Mattmann, 2013). Refocusing on the questions surrounding the KOR-dynorphin system, the latest experimental approaches used to probe this system are introduced.

BEHAVIOR

As previously discussed, behavioral implication is the primary motivator for research into the KOR-dynorphin system. To better understand the relationship between stress, cognition, and behavior, appropriate experimental paradigms to assess behavior are needed. The following behavioral assays assess various cognitive effects associated with activation of the KOR-dynorphin system.

DRL

One cognitive output of interest is inhibitory control. Inhibitory control, or rather the loss-thereof, is associated with various behavioral and mood disorders such as obsessive-compulsive disorder (OCD), anxiety, posttraumatic stress disorder (PTSD), and substance use disorder (Chamberlain et al., 2005; Clauss and Blackford, 2012; Falconer et al., 2008; Jentsch and Taylor, 1999). Stress-induced KOR activation has been shown to increase the behavioral and cognitive symptoms of these disorders, but a direct link between KOR and inhibitory control had not been investigated.

One method to investigate inhibitory control is through the differential reinforcement of low response rate (DRL) task. The DRL task reinforces responses (lever presses) only if an animal withholds responding for a prescribed wait time (Sidman, 1956). After a training period, the wait time is set and consistent across animals and trials. To assess the role of stress and KOR on task behavior, behavioral, pharmacological and genetic perturbations can be used in conjunction with the DRL task (Abraham et al., 2018).

The output from this type of paradigm includes within-trial temporal information and performance as well as longitudinal changes in performance. Classically, DRL data is assessed for distribution of inter-response time (IRT), total number of reinforced responses as well as response efficiency (% reinforced) (McGuire and Seiden, 1980; O'Donnell et al., 2005, Selleck et al., 2015), however little attention has been paid to the IRT distribution apart beyond how it informs response efficiency. In essence, complex temporal information is simplified into a binary output: reinforced or non-reinforced. In order to better quantify impulsivity with the DRL task, there is a computational need to analyze the temporal data with greater resolution of result output.

DELAYED ALTERNATION

Another cognitive output of interest is disruption of working memory. Memory impairment is a symptom in a number of mood disorders (Morris and Cuthbert, 2012). Furthermore, stress is known to have complex effects on working memory through disruption of the inputs to the hippocampus and PFC (Luithi et al., 2008; Weerda et al., 2010). KOR agonism has been shown to disrupt working memory in behavioral tasks, however there is also evidence that KOR

activation can improve memory (Carey et al., 2009; Kuzmin et al., 2006). Therefore, there is a need to better understand the role of KOR in working memory.

One method to investigate working memory is through the operant delayed alternation task. The operant delayed alternation talk reinforces responses (lever presses) only if an animal correctly alternates levers after a subscribed mandatory wait time (Izaki et al., 2001). Similar to the DRL task, the role of KOR can be assessed through pharmacological or genetic manipulation prior to the task.

Also similar to the DRL task, the output from this type of paradigm includes within-trial temporal information and performance and longitudinal changes in performance, as well as lever press patters within and across trials. Classically, delayed alternation data is assessed for percent of correct alternations and perseverance (repetition of the same lever press) for a given trial, however little attention has been paid to the timing of these responses or how the timing and performance change temporally within a single trial. In order to better quantify the type of effects KOR has on working memory, there is a computational need to analyze the finer temporal characteristics of performance in the delayed alternation task.

CIRCUIT

In addition to higher resolution of behavioral outputs, a better understanding of the circuit dynamics underlying those behaviors is necessary. A key component of understanding the circuit to behavior relationship is elucidating the real-time activity of neurons within a given region during behavioral tasks. The following investigative technique can assess various aspects of neural activity at the regional level during behavior.

FIBER PHOTOMETRY

One measure of neural activity is calcium signaling. Calcium ions (Ca^{++}) enter neurons during the firing of an action potential and during synaptic transmission. Therefore, the concentration of Ca^{++} in the cell can be a correlate for neural activity (Yasuda et al., 2004). Genetically encoded calcium indicators (GECIs) such as GCaMP are the state-of-the art for calcium-signaling analysis (Tian et al., 2009). In addition to the monitoring of neural activity, other byproducts of pathway activation can be monitored with similar optically active proteins. For example, administration of opioids has been shown to produce reactive oxygen species (ROS), with potentially detrimental effects (Skrabalova et al., 2013; Schattauer, 2017b). ROS production can be monitored with the optically active protein HyPerRed.

In order to monitor calcium or ROS activity during behavior *in vivo*, fiber photometry can be used. Fiber photometry utilizes implanted optical fibers to capture photon release from optically active proteins to detect broad activity of neurons or pathway byproducts within the implanted region (Cui et al., 2013; Gunaydin et al., 2014). To assess the disruption in circuit dynamics

during behavior, various behavioral assays, pharmacological agents, and genetic manipulation can be utilized prior to or during fiber photometric recording.

The output from this type of recording includes temporal intensity information across the duration of the trial as well as behavioral and pharmacological time points that can be time-locked with the neural signals. The sheer amount of data collected during these types of recordings can cause data handling issues. Classically, fiber photometry data is down-sampled and assessed for change in fluorescence over a baseline or quantified by the occurrence of events consistent with a set of parameters (i.e. signal exceeding a % change threshold or short spikes of increased activity). However, these analyses only scratch the surface of the available information recorded in each session. In order to better unravel the neural activity and outputs contributing to disruptions in circuit dynamics during behavior, there is a computational need to improve the data handling techniques and breadth of analyses available to researchers.

OPTRODE

While fiber photometry provides high temporal resolution, its spatial resolution is limited to a single brain region. To truly investigate circuit dynamics, there is a need to develop *in vivo* multi-site recordings tools and analysis techniques. The proposal and design of such a tool is documented in Appendix A.

CELLULAR

Lastly, proper interpretation of neuronal activity and outputs is reliant on an understanding of the cellular mechanisms underlying those endpoints. As previously discussed, activation of KOR initiates two signaling cascades resulting in variable physiological effects. Much effort has been devoted to quantifying the mRNA and protein levels of various players in those signaling cascades, but these methods neglect an important temporal variable (Schattauer et al., 2012). The following investigative technique can assess various aspects of cellular mechanisms while maintaining temporal resolution.

SINGLE CELL HYPERRED IMAGING

In order to monitor the temporal components of cellular mechanisms, real-time microscopy can be coupled with optically active proteins like those used in fiber photometry. Real-time cellular microscopy records a series of images over the course of an experimental time period to track changes over time. This technique allows researchers to administer a pharmacological agent and observe the cellular effects in real-time.

The output from real-time cellular microscopy includes a series of images with pixel intensity information captured over time. Classical analysis of cellular microscopy requires selection and numbering of cells by hand prior to computer-facilitated quantification. This process is time

consuming and leaves room for biasing and arbitration of results (Ninomiya et al, 2016). In order to efficiently and appropriately analyze real-time cellular microscopy images, there is a computational need to automate the cellular identification, tracking, and quantification steps.

CURRENT UNDERSTANDING

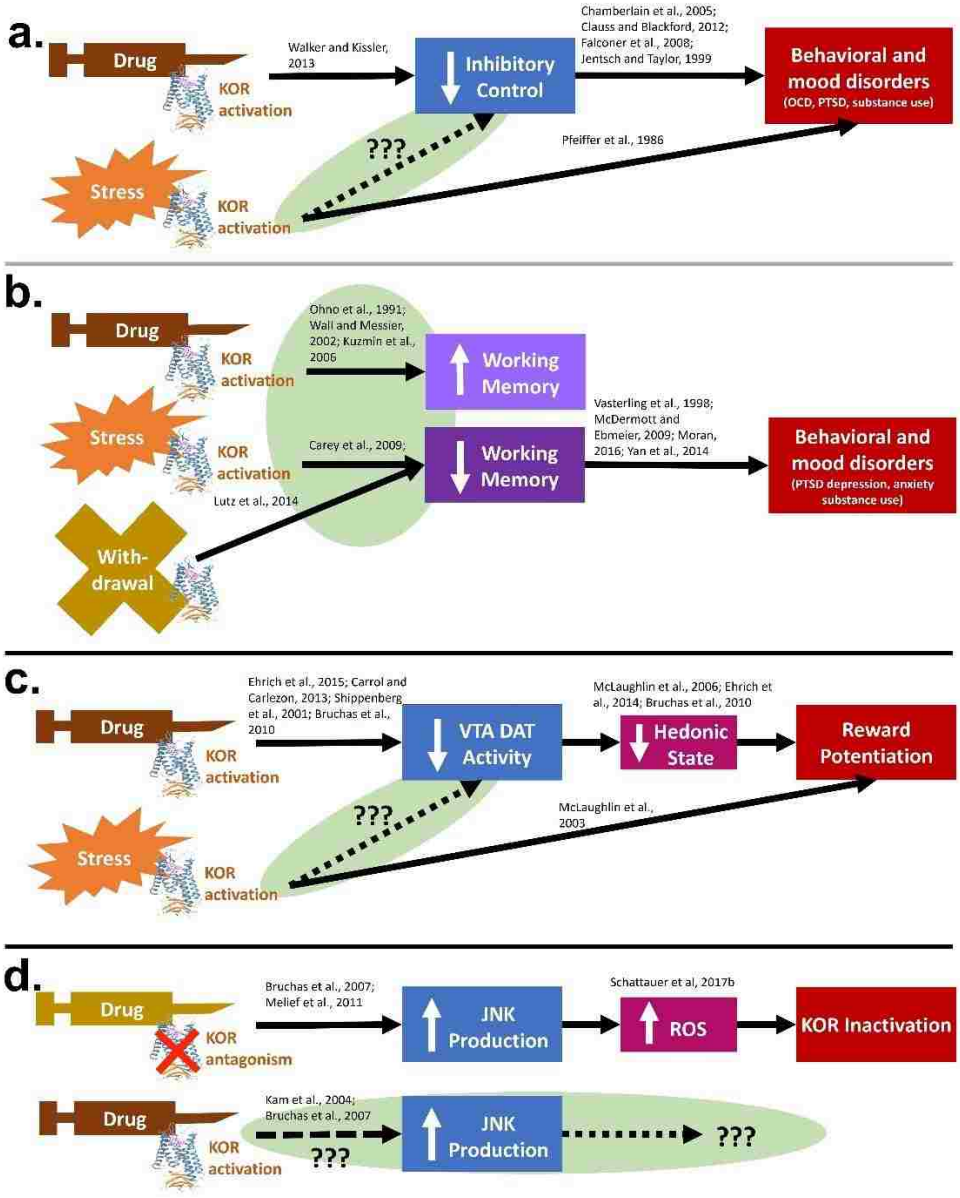

Figure 2: State of Understanding Prior to Presented Work

Schematics depicting the relevant understanding of behavioral **(a-b)**, circuit **(c)**, and cellular **(d)** level relationships between KOR activation and respective outputs. Dashed lines and ovals depict the gaps in current understanding that will be investigated in Chapters 2-4.

Note: Primarily intended as a reference for the information presented in the Chapters 2-4, compare to Figure 22.

KEY PHARMACOLOGICAL AGENTS

As a reference for the content in future chapters, Table 1 summarizes some of the key pharmacological agents used in the experimental approaches to probe the KOR-dynorphin system.

Compound	**κ Activity**	**General Effect Through κ**
Dynorphin	Endogenous agonist	Analgesia, dysphoria, prodepressive,
Morphine	Weak agonist	Analgesia
Salvinorin	Agonist	Hallucination, dysphoria, anxiogenic
U50,488	Agonist	Dysphoria, reinstatement
Nalfurafine	Gβγ biased agonist	Analgesia without dysphoria
Nor-BNI	Antagonist	Anti-depressant,
Buprenorphine	Antagonist	Anti-depressant
Naloxone	Antagonist	Precipitated withdrawal, hyperalgesia
MJ33	PRDX6 selective inhibitor	Prevents norBNI inhibition, Reduced tolerance to morphine

Table 1: Key Pharmacological Agents

CHAPTER 2 – BEHAVIOR

OVERVIEW

Chapter 2 introduces the importance of appropriately regulated motivated behavior and the factors underlying this branch of behavior. This information details the biological, environmental, cognitive, and emotional inputs that are integrated to produce a behavior, especially within the context of risk and reward valuation. Stemming from this context, the effect of stress on the production of both adaptive and maladaptive behaviors is described. Following, two aspects of motivated behavior that can be disrupted by stress – inhibitory control and working memory – are explained within both a physiological and pathological context. Finally, the role of KOR, on the aforementioned stress effects is described.

This chapter describes two behavioral assays that can be used to investigate KOR-mediated stress effects on inhibitory control and working memory: the differential reinforcement of low response rate (DRL) task and the operant delayed alternation task, respectively. The current gaps in knowledge leading to the research questions being investigated are explained. Then, the experimental and computational methods used to investigate these questions are described in detail, with an emphasis on novel computational approaches to analyzing the data produced.

Finally, the experimental results discovered through these assays are presented within the context of KOR-mediated implications on motivated behavior.

BACKGROUND AND SIGNIFICANCE

Just as a computational system integrates a series of inputs to generate an output, biological systems integrate environmental factors with internal signaling within their genetic framework to produce a functional output – namely, behavior. Motivated behavior is the critical output that determines an organism's survival, fitness, and well-being. Underlying motivated behavior is a complex, and sometimes unstable, balancing act between risk and reward. Every behavioral action has an associated risk, from basic energy expenditure to imminent survival consequences like predation. Thus, the greater the risk, the greater the reward must be to pursue the behavior.

Motivated behavior is the dynamic product of four major factors: biology, environment, cognition, and emotion (Petri, 2005). Biology hardwires some innate behaviors such as reflex responses, arousal, aspects of social interaction, and feeding (McDougall, 1970; Lorenz, 1950; Tinbergen, 1951). However, biology also has more subtle effects on how an individual responds to environmental stimuli and their susceptibility to cognitive and emotional disruptions (Fraga et al., 2005).

Environmental stimuli have a major effect on behavior. Behaviors learned through observation or via classical or instrumental conditioning are typically not exhibited unless there is a motivation for that behavior (Lieberman, 2004). This motivation can come from a negative consequence – risk (e.g. punishment) – which reduces the probability of a future action, or from a positive consequence – reward – which increases the probability of a future action (Watson and Rayner, 1920). However, motivation can become destabilized when the risk or reward values of an action are unclear, i.e. when an individual does not know whether an action will generate more reward than risk, or when an individual's reward circuitry is disrupted (Pavlov, 1960). This reward valance ambiguity can result in maladaptive behaviors. As discussed in Chapter 1, an environmental stressor is an aversive experience and chronic stress can skew the risk-reward valuation of a situation. For example, animals who are in a prolonged stressful situation without the ability to escape will exhibit "learned-helplessness" – they will stop trying to escape even when an escape option is made available (Overmier and Seligman, 1967).

To interpret the risk-reward ratio of a situation and act on or pursue a previously learned behavior, cognition comes into play. One theory surrounding the cognitive component of motivation is expectancy theory – the evaluation of previous successes and failures, present expectations of success, and the reward value of a given situation (Atkinson and Birch, 1978). In order to evaluate a situation and develop an appropriate reward expectation for an action, an individual must properly integrate reward signaling, aversion and reward processing, memory, and decision making, requiring coordination of the VTA, NAc, hippocampus, and PFC (see Figure 1). Since stress is known to produce physiologic changes in these regions, stress, and its associated physiologic effects, will affect cognition and decision making in a learned task.

Following from expectancy theory, prior experiences affect not only the cognitive, but the emotional appraisal of the current situation (LeDoux, 2000; Lazarus, 1982). Additionally, changes in emotional state will affect the subsequent reward appraisal and resultant action in a situation. As discussed in Chapter 1, chronic stress can result in maladaptive changes in emotional state which, by expectancy theory, can deregulate the reward appraisal of a situation resulting in behaviors with averse consequences such as learned helplessness or loss of inhibitory control (Moore et al., 2008; Tice et al., 2001, Moeller et al., 2001).

Situationally appropriate behavior is a critical part of survival, thus developing a better understanding of the factors that motivate or disrupt behavioral adaptation is critical. Since the factors driving behavior (biology, environment, cognition, and emotion) are individually complex their interactions become exponentially complex in an uncontrolled setting. To investigate how environmental responses, such as stress, affect the cognitive and emotional regulation of behavior, controlled and targeted experimental paradigms with appropriate analysis methods are necessary. This chapter introduces two behavioral paradigms that target specific aspects of the relationship between stress, cognition, and behavior, details the inputs and outputs under investigation, and describes computational methods to better utilize and interpret the data produced.

DRL

MOTIVATION

As introduced, behavior is regulated by assessment of the risk-reward ratio of a situation. While the obvious outcome of this assessment is the performance of an action, an equally important outcome is the inhibition of action. If a behavior is learned or expected to result in more risk than reward, the appropriate decision would be to withhold action. However, maladaptive mechanisms can result in the loss of this inhibitory control (Arce and Santisteban, 2006; Kirkpatrick et al., 2013).

Inhibitory control requires two broad functions: selection and stopping (Anderson and Weaver, 2009). Selection refers to the decision between multiple competing actions in response to a given set of cues. As a basic example, the appropriate selection between 'fight' and 'flight' has critical implications on an organism's survival. A hyper-aggressive individual may impulsively select to fight in an inappropriate situation, resulting in maladaptive consequences. Stopping refers to the preempting of an action that would have undesirable consequences, such as not catching a falling object if that object is realized to be a hot pan. Another critical component of inhibitory control is timing (Brown et al., 2015). A basic example of timing related inhibitory control is the Stanford marshmallow task in which a less preferred reward could be attained with immediate action, but the inhibition of action until a later time point would grant a greater reward (Mischel et al., 1972). The ability to select a preferred course of action, stop an immediate action, and appropriately time an action to achieve the desired outcome are critical aspects of inhibitory control and adaptive behavior.

The loss of inhibitory control is associated with various behavioral and mood disorders such as obsessive-compulsive disorder (OCD), anxiety, posttraumatic stress disorder (PTSD), and substance use disorder (Chamberlain et al., 2005; Clauss and Blackford, 2012; Falconer et al., 2008; Jentsch and Taylor, 1999). Environmental stressors are known to amplify behavioral symptoms of these disorders including loss of inhibitory control (Fornaro et al, 2009). Stress-induced KOR activation has been shown to increase the behavioral and cognitive symptoms of these disorders and pharmacological KOR activation has been shown to degrade response inhibition (Walker and Kissler, 2013). However, a direct link between stress-induced KOR activation and a loss of inhibitory control had not been investigated.

EXPERIMENTAL APPROACH

To investigate the timing and stopping aspects of inhibitory control, a time-dependent reinforced task paradigm can be used. A validated paradigm for testing in rodents is the differential reinforcement of low response rate (DRL) task. The DRL task requires animals to wait a set period of time between responses in order to receive the reward (Sidman, 1956). If the animal does not wait the prescribed time, the action is not rewarded and the time resets. A typical

response task used for mice is a nose poke with the reward being a sucrose pellet. In this task, the behavioral decision of nose-poking requires the mouse to integrate the passing of time with the evaluation of the expected reward – sucrose – and the potential risk – resetting the wait period.

The goal of this assay was to assess the role of stress and KOR on a learned, time dependent behavior to inform KOR-dependent disruptions of inhibitory control. Beginning with the behavioral phenomenon observed, forced swim stress paradigms, systemic KOR agonism, local KOR antagonism, and genetic excision of KOR or deletion of GRK3/p38 MAPK were used to determine regional characteristics, neuronal population effects, and the cellular signaling mechanisms behind the behavioral effects observed (Abraham et al., 2018).

METHODS

The DRL task consists of 3 specific periods in the experimentation process: behavior acquisition, behavior stabilization, and a test period. During behavior acquisition, the mice learn the task through trial and error of nose poking for their reward. During the behavior stabilization period, the learned behavior of the mice is reinforced through repeated sessions and monitored for consistency. The test period generally consists of 2 test days, a control day followed by a stress or drug trial test day.

The DRL operant chambers have two nose-poke holes: active and inactive. The mice were first trained to discriminate between the active and inactive holes. Following, mice were trained in a 60-min DRL procedure (Horwood et al, 2001). During each session, a single nose poke led to food reward delivery after which the mouse was required to withhold responding for a specified wait period – 15 seconds. If the mouse nose poked before the end of the wait period, the wait time reset and no reward would be administered. After a training period, test sessions included either systemic U50,488 treatment; nalfurafine treatment, repeated forced-swim stress (rFSS); microinjection of nor-BNI into the PFC, DRL, or VTA; or conditional or global knockout of KOR or GRK3/p38 MAPK followed by U50,488 administration.

The software connected to the operant chamber recorded timestamps for active nose pokes, reinforced nose pokes, head entries into the pellet hopper, wait period resets, and inter response times (IRTs). The data was exported as a .txt file.

COMPUTATIONAL APPROACH

Classically, DRL data is assessed for distribution of inter-response time (IRT), total number of reinforced responses as well as response efficiency (% reinforced) (McGuire and Seiden, 1980; O'Donnell et al., 2005, Selleck et al., 2015). This type of analysis reduces the complex temporal information into a binary output: reinforced or non-reinforced. Changes in performance efficiency can be explained by a number of factors such as disruption of memory formation or retrieval, disruption in time perception, task distraction, or loss of inhibitory control. In order to

better dissociate and quantify the loss of inhibitory control from other factors, there is a computational need to analyze the temporal dimension of the data with greater resolution of result output.

Specific effort was devoted to quantifying 'burst responses' – a non-reinforced response within 1 second of a previous response. Responses occurring within such a short time period are unlikely to be cause by disruption in time perception, memory formation, or distraction. Thus, these burst responses are most likely attributed to deficits in inhibitory control (Selleck et al., 2015).

HANDLING

The DRL system generates a large amount of data per animal, per run. The times of every reinforced nose poke, reset response, head entry, inactive nose poke, and the inter response time between each nose poke are recorded. A given run can generate between 500 and 3000 pieces of data per animal. With approximately 7 runs per day, with 8 animals per run, over the course of 30 test days, there was approximately 1 million raw data points generated.

In order to read each file into MATLAB (version R2016a) and organize both by date and by cohort, the files were named in an iterative format. Therefore, the DRL .txt files were named by their date followed by their cohort and placed into a single destination folder (ex. "20150326_drl5.txt"). A date vector with integer values corresponding to the dates of the test days for each cohort was declared (Appendix II, DRL Code, lines 6-8). The data were then extracted from the destination folder using `textread`. An outer for-loop loops through the date vector to pull out the integer value at the given index (lines 10-12). A string can then be concatenated from the integer values and the cohort name to read the file name from the folder (lines 21-22).

Once the file was read into MATLAB, a search algorithm (`strfnd`) was used to find the 5 subheadings within the file that precede the related data. The line number at which the given subheading was found within the file was then stored to a vector (lines 24-25). For example, the sub header "ActiveNosePokeTimes(sec)" preceded all of the number values representing the time of an active nose poke during the testing period. The line number where the text "ActiveNosePokeTimes(sec)" was found in the file would be stored to the vector `IndexActNP`.

PROCESSING

A matrix was used to store the time data for each subheading within the file. I will be referring to the Active Nose Poke matrix code (lines 43-64), however the method is the same for each subheading. Each file contains all of the animals for that run (up to 8 animals), additional for-loops with `i` values denoting the number of animals in the file is used to parse through the index vectors instantiated in the previous method (line 43). Within each of the `i` for-loops, a start and last variable are pulled from the index vectors to denote the line numbers between which the data under the subheading for the specified animal is contained (lines 44, 45). The data information

between those 2 maximal values is stored to a temporary cell (line 46). The length of that cell is then checked against the length of the preexisting matrix storing the data cells from the previous animals in the run. If the length of the new cell is shorter than the preexisting matrix, the appropriate number of zero entries are concatenated to the end of the additive cell. If the length of the matrix is shorter than the additive cell, the appropriate number of zeros are added to the bottom of each column of the matrix (lines 47-62). The new cell is then concatenated column-wise to the pre existing matrix (line 63).

At the end of each `i` loop, a matrix containing as many columns as animals in the cohort will have the specified subheading data times in the columns, with zero padding at the bottom of each column to fill in the whole matrix. I used MATLAB's `eval` function to save each of the 5 matrices generated per `r` loop so that the date was iteratively inserted into the name of the matrix (lines 164-168). For example, the temporary `ActiveNP` storage matrix would be renamed and stored as `ActiveNP_0204` and so on for each subheading and each test day.

From the data collected, there were 6 critical pieces of information required to answer the research questions posed. Collected from the data were all of the inter response times (amount of time between 2 consecutive responses) for each animal, as well as the total number of reinforced responses, error responses, and head entries. Finally, computed from the tallied totals were the total number of responses and the percent error of each animal's responses.

To extract the above information, a new date vector for-loop was instantiated (line 176). The `eval` function was then utilized again to copy the `resets`, `reinNP`, and `headEnt` matrices for the given day to temporary storage matrices (lines 179-181). A nested for-loop, `a`, with number of iterations equal to the number of animals in the cohort was instantiated to parse animal specific information from the columns of the data matrices (line 183). Each column of the given matrix was isolated into a vector (line 184), and any zero padding at the end of the vector that was added in the previous method was removed (line 185). The length of that vector was then stored to a variable (line 186) which was then stored to a matrix coordinate with rows corresponding to the date and columns corresponding to the animal (line 187). For example, the number of error responses on the 6th test day for the 9th animal in the cohort would be saved to the 6th row in the 9th column of the `numResets` matrix.

ANALYSIS

PERCENT ERROR

The percent error calculation took the total number of error responses for a given animal and divided that by the sum of the number of error responses and the number of reinforced responses. This value was then stored into the `percentError` matrix in the same fashion as described above (line 199). Finally, to compute the total response number matrix, the error response and reinforced response matrices were summed together (line 203).

BURST ANALYSIS

The final research questions pulled directly from the data were related to the concept of burst responses. A burst response was defined to be any response occurring within 1 second of the preceding response. A series of burst responses are grouped into clusters. The burst responses were then split into 2 categories: bursts/clusters following an error response and bursts/clusters following a reinforced response.

Using another date for-loop and the `eval` function, temporary matrices were instantiated and stored with the reset times, reinforced nose poke times, and inter response times for the specified date (lines 205-215). A for-loop was used to parse through the inter response times for each animal. Any zero padding at the end was turned into a series of 9's. Any inter response time above 1 second was stored as a 0 (negative for a burst response) and any inter response of 1 second or under was stored as a 1 (positive for a burst response (lines 217-226). A series of 3 if-else statements nested within a while-loop were used to develop a second binary vector which stored a 1 for a reinforced response or a 0 for an error response for every response given by an animal (lines 234-254).

A final series of nested if-then statements, for-loops, and while-loops compared the binary burst vector with the binary correct response vector. 4 different types of burst responses were recorded: a 1 denotes a burst after a correct response, a 2 denotes a burst after an error response, a 3 denotes a burst after a correct burst (signifying a successive reinforced cluster response), and a 4 denotes a burst after an error burst (signifying a successive error cluster response) (lines 261-304). This burst analysis was completed for each animal and saved to a matrix with the columns representing the analysis results for each animal. The matrix was then saved with the date via the iterative naming process described in method 2 (line 305).

COMMUNICATION

RASTER PLOTS

A raster plot is a way of communicating discrete events (such as a nose poke) over a period of time. Once all of the data was extracted and stored in MATLAB matrix format, the data needed for the raster plot had to be isolated and organized into a plottable vector format. A vector containing the animal numbers used for the plots was defined for each cohort (lines 488-490). To compare all of the animals and runs on a single plot, the saline and test days were separated on the y-axis by 0.5 units (lines 492, 493). For each cohort of animals, a for-loop was used to loop through the matrix stored from the Data Matrix Storage step to isolate the time data for each animal of interest (line 495). Two vectors were initialized containing the reinforced and reset times for the animal of interest (lines 497, 498). Nested for-loops were then used to remove the empty indices concatenated on the vectors during the Data Matrix Storage step (lines 500-512).

Since all of the time data for a single run would be plotted at a single y-axis value, a vector of the same length as the time data vectors was instantiated with 1's (lines 514-515). For-loops were then used to re-instantiate the y-vectors with the appropriate placement on the y-axis (lines 517-523). The reinforced response were then plotted as a dot at the appropriate time indices and the reset responses were plotted as a "+" sign (lines 525-528).

RESULTS

The computational methods described above allowed for efficient and robust handling, processing, analysis, and communication of the DRL data. Classical presentation of DRL data includes IRT binning and binary performance plots. However, those summaries lose much of the temporal information of an animal's behavior. To visualize the complete response activity of an animal during a test session, raster plots were used. A representative raster plot of a male mouse with saline and U50,488 pretreatment is shown (Figure 3, from Abraham et al., 2018). This type of data presentation displays the total number of responses, ratio between reinforced and non-reinforced responses, all of the IRTs, and temporal patterns across the trial. For example, under U50,488 treatment, it is clear that the number of non-reinforced responses increased and the number of reinforced responses decreased, which is classically summarized with a bar graph. However, it is also apparent that the response frequency during the first 30 minutes of the U50,488 session is lower which suggests information about the temporal effects of U50,488 – information not captured with classical presentation methods.

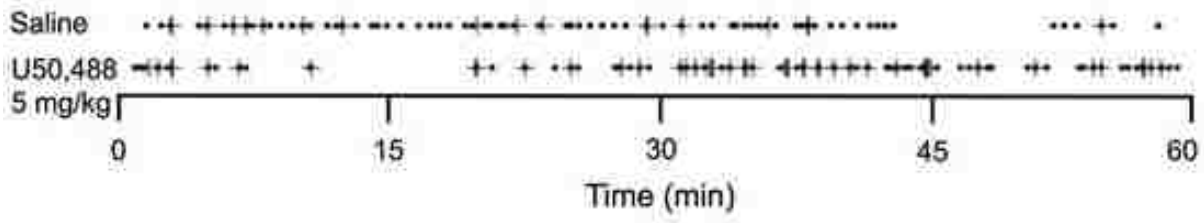

Figure 3: Raster Plot of DRL Behavior with Saline and U50,488 Pretreatment

Raster plot of a sample data set. Responses from a single male animal during a 60 min DRL sessions with saline (top) or U50,488 (bottom) pretreatment. Closed circles represent reinforced responses and '+' symbols represent nonreinforced responses. From Abraham et al., 2018.

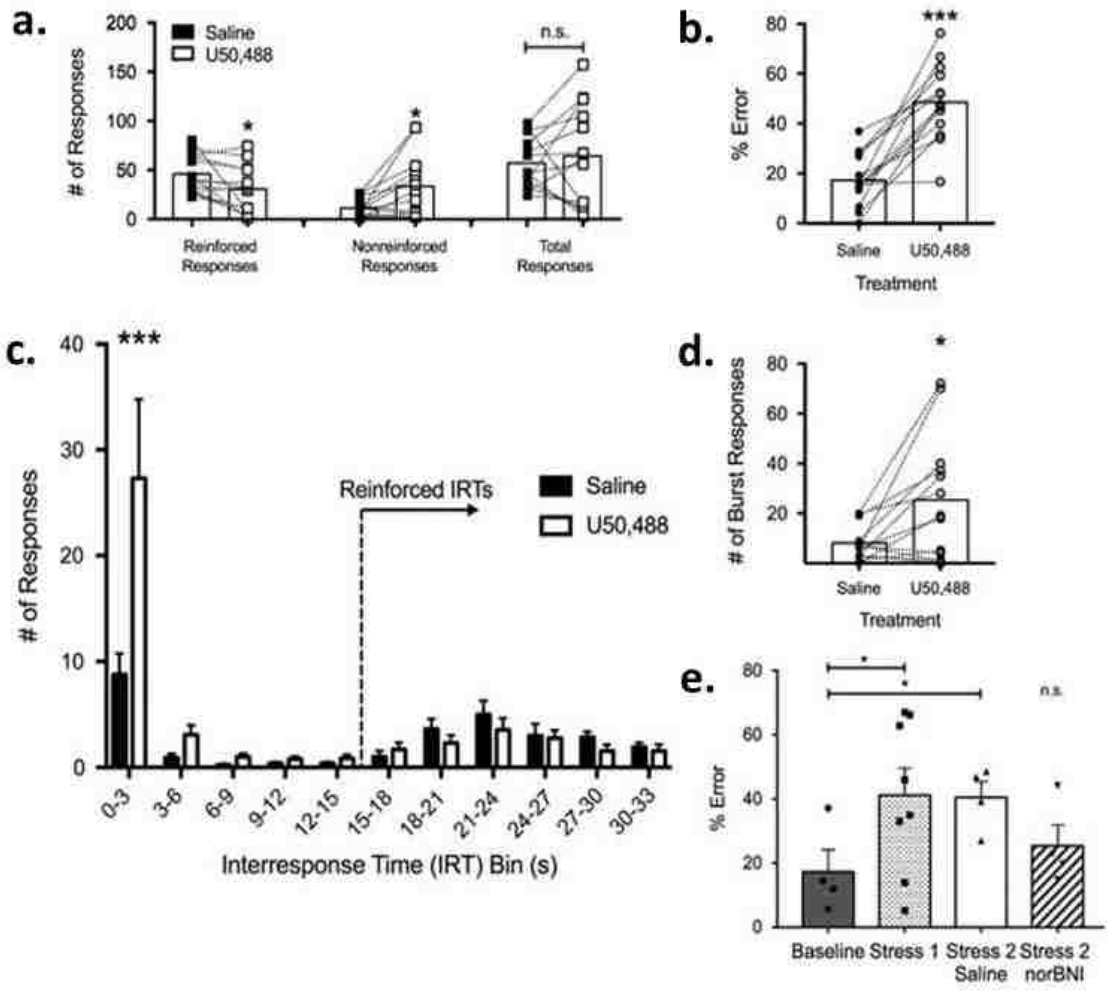

Figure 4: System KOR Activation Produced Disruptions in DRL Performance

(a) KOR activation by U50,488 decreased the total number of reinforced responses and increased nonreinforced responses without altering total number of responses. The number of responses made by an individual mouse during a 60 min period following saline pretreatment were compared with responses following U50,488 pretreatment by a paired t-test (*po0.05). **(b)** KOR activation significantly increased percent error during the DRL session. **(c)** A histogram of interresponse times showing the number of responses per 3 s bin. U50,488 increased responses occurring within 0–3 s of the previous response. **(d)** KOR activation significantly increased the number of burst responses (an additional response ≤1 s after the previous response). **(e)** Repeated forced-swim stress increased percent error. Pretreatment with nor-BNI (KOR antagonist), but not saline, blocked increases in percent error following a second exposure to repeated forced-swim stress compared with baseline. Error bars indicate SEM. *P<0.05; ***p<0.0001; NS, not significant. Adapted from Abraham et al., 2018.

As interpreted from Figure 3, the animal's response frequency is lower during the first 30 minutes of U50,488 treatment compared to control treatment. This is potentially due to U50-induced locomotor suppression (Paris et al., 2011). However, this suppression of activity at the beginning of the trial does not affect the total number of responses across the trial (Figure 4.a; adapted from Abraham et al., 2018). Importantly, U50,488 treatment does increase the percent error during the task (Figure 4.b), due in part to an increase in burst responses (Figure 4.d).

Additionally, pre-exposure to a stress paradigm increases the response error, but that response is blocked by KOR-antagonist nor-BNI, suggesting that the stress-induced behavioral disruption is KOR-dependent.

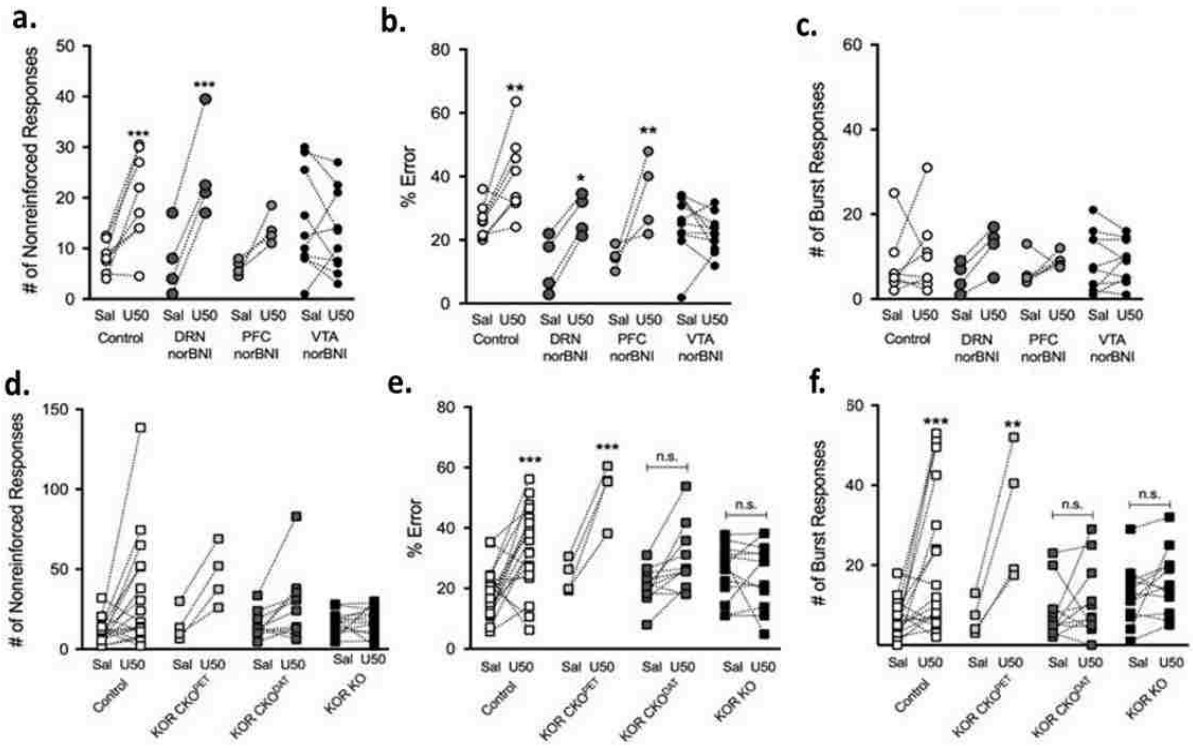

Figure 5: KORs in VTA Dopamine Neurons Are Required for KOR-Mediated DRL Disruptions

(a) There was a significant increase in the number of nonreinforced responses in Control and DRN/nor-BNI, but not VTA nor-BNI or PFC/nor-BNI, following U50,488 treatment. **(b)** U50,488 caused a significant increase in percent error in Control, DRN/nor-BNI, and PFC/nor-BNI mice, but not VTA/nor-BNI-injected mice. **(c)** There was a significant effect of U50,488 treatment on the number of burst responses, but no significant interaction between treatment and brain region. **(d)** U504,88 treatment significantly increased the number of nonreinforced responses but there was no significant interaction between treatment and genotype. KOR activation significantly increased percent error **(e)** and burst responses **(f)** in Control and KOR CKOPET mice, but not KOR CKODAT or KOR KO mice. Error bars indicate SEM. *P<0.05; **p<0.01, ***p<0.0001. Adapted from Abraham et al., 2018.

At the circuit level, micro-injections of nor-BNI in the VTA, but not the PFC or DRN, block U50-induced increases in percent error (Figure 5.b; adapted from Abraham et al., 2018). However, the increase in percent error in the PFC is due to a non-significant increase in non-reinforced responses (Figure 5.a) and not an increase in burst responses (Figure 5.c), potentially suggesting that the PFC plays a role in the loss of inhibitory control observed through the burst response output. Additionally, both global KOR knockout and conditional knockout of KOR in dopaminergic (DAT) neurons blocked the U50-induced increases in both percent error (Figure 5.e) and burst responding (Figure 5.f), suggesting that dopamine neurons are required for KOR-mediated disruptions in DRL performance and inhibitory control. Lastly, conditional knockout of p38 MAPK from dopamine neurons (Figure 6.a, from Abraham et al., 2018) and global deletion

of GRK3 (Figure 6.b) does not prevent KOR-mediated increases in percent error, and G-biased KOR agonist nalfurafine significantly increases percent error (Figure 6.c). These results suggest that KOR-mediated DRL disruptions are arrestin independent.

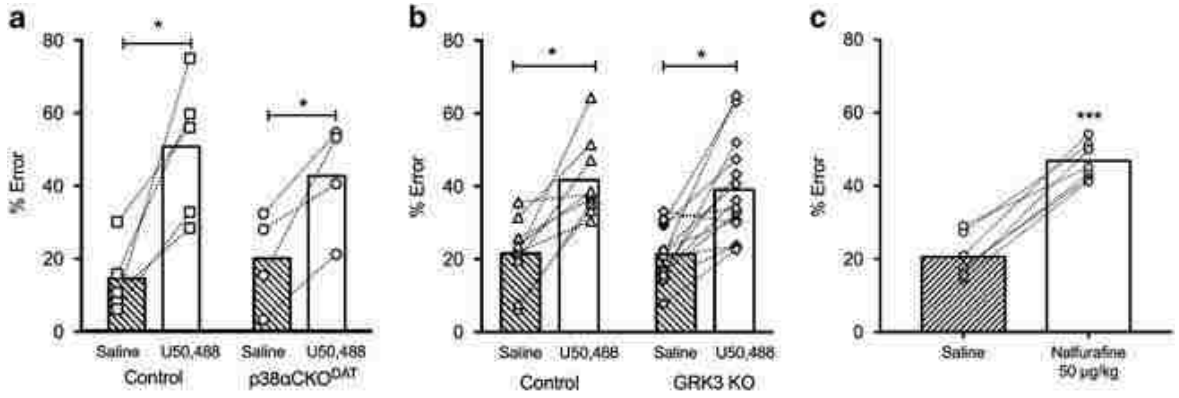

Figure 6: KOR-Mediated DRL Disruptions are Arrestin Independent

(a) Conditional deletion of the p38α MAPK in dopamine neurons does not prevent KOR-mediated increases in percent error in the DRL task. **(b)** Global deletion of the GRK3 does not prevent KOR-mediated increases in percent error in the DRL task. **(c)** The G-biased KOR agonist nalfurafine significantly increases percent error in the DRL task, indicating that DRL disruptions are arrestin independent. Error bars indicate SEM. *P<0.05, ***p<0.0001.

Cumulatively, these results demonstrate that KOR-activation, either by stress or pharmacological means, plays an important role in the disruption of a time-dependent behavioral inhibition task. The VTA was required for these behavioral disruptions suggesting that reward value and reward expectancy are important in the disruption of inhibitory control (Schultz et al., 1997; Hart et al., 2014).

My results from this analysis were published: Abraham, A.D., Fontaine, H.M., Song, A.J., Andrews, M.M., Baird, M.A., Kieffer, B.L., Land, B.B., Chavkin, C. (2018) Kappa opioid receptor activation on dopamine neurons disrupts behavioral inhibition. Neuropsychopharmacology, 43:362-372. [PMC5729556]

DELAYED ALTERNATION

MOTIVATION

As demonstrated by the DRL assays, the VTA's encoding of reward valuation and reward expectancy play an important role in an inhibitory control task. This further supports the involvement of expectancy theory in behavioral control and motivated decision making. As introduced previously, expectancy theory involves the evaluation of previous successes and failures, present expectations of success, and the reward value of a given situation (Atkinson and Birch, 1978). In order to appropriately place an expectation on a certain action, memory of previous experiences and the events leading up to the current decision are necessary.

Generally, there are three types of memory that can be integrated to form decisions and inform behavior: long-term, short-term, and working memory (Cowan, 2009). Long-term memory is the large store of knowledge of prior events. Short-term memory refers to a duration and capacity restricted cache of information about recent or recently recalled events. Short-term memory may involve conscious or unconscious information encoded by presently active cell-assemblies (Hebb, 1949). Working memory is often considered a subset of short-term memory and is involved in the conscious thoughts used to plan and carry out a behavior (Miller et al., 1960). Working memory is involved in cognitive decision making and is prone to disturbance and distraction (Unsworth and Engle, 2007). Playing chess is an example – to become good at chess, a player must integrate memories from previous games (long-term memory) with information about recent moves (short-term memory) and compute the potential outcomes of a given set of potential next moves (working memory).

Impaired working memory is associated with various stress-related psychiatric disorders including PTSD, depression, anxiety, and substance use disorder (Vasterling et al., 1998; McDermott and Ebmeier, 2009; Moran, 2016; Yan et al., 2014). Stress is known to have complex effects on working memory through disruption of the inputs to the hippocampus and PFC (Luithi et al., 2008; Weerda et al., 2010). Furthermore, both stress-induced activation of KOR and protracted withdrawal from opiates has been shown to cause deficits in learning and working memory (Carey et al., 2009; Lutz et al., 2014). However, there is conflicting evidence around the role of KOR in these deficits, with KOR agonism being shown to improve working memory in some studies (Ohno et al., 1991; Wall and Messier, 2002; Kuzmin et al., 2006).

EXPERIMENTAL APPROACH

Behavioral paradigms to investigate working memory require both a decision that is determined by a prior event and a delay between the prior event and the decision. A validated paradigm for testing working memory in rodents is the operant delayed-alternation task (Rossi et al., 2012; Figure 7). The delayed-alternation task requires an animal to make a free response, wait a set period of time, then make an alternation contingent response, i.e. respond differently from the first response (Izaki et al., 2001; Dudchenko, 2004). The animal is only rewarded if their second response is an alternation from the first response. A typical response task for a mouse is a lever press with the reward being a sucrose pellet. In this task, the decision of which lever to press requires the mouse to integrate their previous decision with the evaluation of the expected reward – sucrose – and the potential risk – resetting the trial.

The goal of this assay was to assess the effect of PFC KOR and opiate withdrawal on working memory. Beginning with the behavioral task, forced swim stress paradigms, systemic KOR agonism with and without local KOR antagonism, and morphine withdrawal were tested to observe their effects on delayed-alternation timing and performance.

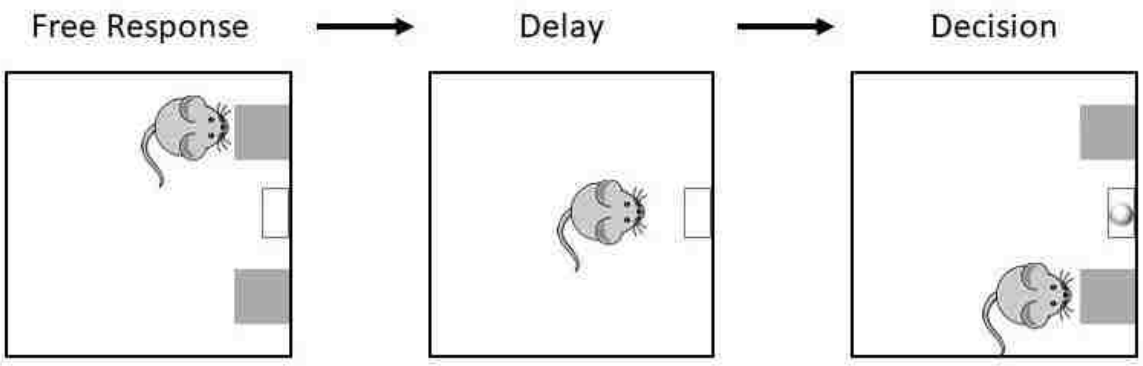

Figure 7: Operant Delayed-Alternation Task Paradigm

To start the trial, the animal is presented with both levers and allowed to make a free response choice (Left). The levers are then retracted for a set wait period (Middle). At the end of the wait period, the levers re-extend, and the animal can make their decision (Right). If the animal chooses the opposite lever as their free choice, they are rewarded with a food pellet.

METHODS

The operant delayed-alternation task consists of 3 specific periods in the experimentation process: behavior acquisition, behavior stabilization, and a test period. During behavior acquisition, the mice learn the task through trial and error of lever pressing for their reward. They are then trained to alternate lever presses for reward. During the behavior stabilization period, the learned behavior of the mice is reinforced through repeated sessions and monitored for consistency. The test period either consists of 2 test days, a control day followed by a stress or drug trial test day, or longitudinal tracking of performance over multiple days.

The delayed-alternation operant chambers have two levers. The mice begin in a fixed ratio (FR1) task where every lever press results in delivery of a sucrose pellet. They then move into 60 minute delayed-alternation conditioning starting with 2-second, then 5-second, then 10-second wait periods. During the delayed alternation conditioning, both levers would be presented, the animal had free choice for their first press. Both levers would then retract during the prescribed wait period with the house light remaining on. When the wait period was over, the levers would come back out and the mouse would make its choice: if the mouse chose the same lever, there would be no reward; if the mouse chose the opposite lever, a sucrose pellet would be delivered. The second lever press concludes the trail – the house lights would turn off, and the levers would retract for 20 seconds during the intertrial period (adapted from Yin, 2010). After a training period, test sessions included either systemic U50,488 treatment in mice with ACSF (control) or nor-BNI microinjection in the PFC; repeated forced-swim stress (rFSS) in mice with ACSF or nor-BNI microinjection in the PFC; or mice with increasing doses of morphine over a 5-day period followed by a 5 day withdrawal period.

The software connected to the operant chamber recorded timestamps for right and left lever presses, correct and incorrect alternations, head entries into the food tray, and delay period start

and end. The data was exported as a .txt file.

COMPUTATIONAL APPROACH

Classically, delayed alternation data is assessed for percent of correct alternations and perseverance (repetition of the same lever press) for a given trial, however little attention has been paid to the timing of these responses or how the timing and performance change temporally within a single trial. In order to better quantify the type of effects KOR has on working memory, there is a computational need to analyze the finer temporal characteristics of performance in the delayed alternation task.

Specific effort was devoted to quantifying the latency of a response (the amount of time after the levers are presented and the alternation choice), the correlation between latency and performance, and within session time binning to observe performance changes within a given session.

HANDLING

The delayed-alternation system generates a large amount of data per animal, per run. The times of every lever press, reinforced and non-reinforced response, head entry, and delay period is recorded. A given run can generate between 1000 and 5000 pieces of data per animal. With approximately 3-8 runs per day, with 2 animals per run, over the course of 94 test days, there was approximately 3 million raw data points generated.

In order to read each file into MATLAB (version R2017a) and organize both by date and by animal, the files were named in an iterative format. Therefore, the files were named by their date followed by their box number and placed into a single destination folder (ex. “20181206cda5”). A folder path was defined, the number of files in the directory was extracted using `load`, `dir` and `numel` and a for-loop was used to extract data from all the files (Appendix II, Delayed Alternation Code, lines 7- 23). A regular expression (`regexp`) was used to load each file as whitespace (`\s+`) and new line (`\n`) delimited (lines 26-27). The raw data was loaded into a cell array and `NaN` values were used to maintain column-wise structure (lines 29-41).

Once the file was read into MATLAB, a search algorithm (`find`) was used to find the 4 data heading within the file that precede the related data. The line number at which the given subheading was found within the file was then stored to the `datalocations` entry in the structured variable for that given file (lines 44-47). The run date was then extracted from the top of the file and stored into the `date` entry of the structured variable (lines 49-51). Finally, the number of boxes recorded in the file was determined by the number of data headings and stored in the structured variable (lines 53-54).

The vector data under each data heading (B, E, and T) was then extracted by evaluating the first line under the given heading and the last line above the next heading, the data was stored in the

structured variable as `parsedData` under the given file, box number, and data heading (lines 56-91). The total number of left and right presses, correct and incorrect alternations on each lever, and pellets delivered was extracted from the B data and stored into the structured variable under `totals` in the given file and box number (lines 93-112).

Animal ID numbers were extracted from the file name using `regexp` and stored in the structured variable under the given file in the `animalID` entry (lines 113-124). If no ID number was present in the file name, that file was flagged and the file name was stored in the structured variable to be checked (lines 112, 118-119, 126-128). All of the data was then restructured to be organized by date as opposed to file name into the `sortByDate` structure (lines 156-205) (Figure 8). Finally, the dates for each run were translated into number of days from the first delayed alternation training day (10/29) and stored in the vector `days` (lines 210-214).

Finally, the event and time stamp information were extracted from `parsedData` and stored into the `sortByDate` structure. The index of every right lever press, left lever press, correct alternations right, correct alternation left, incorrect alternations right, incorrect alternations left, delay start, and delay end were stored into vectors (lines 514-521). Those indices were then coded into a vector, `press_latency_vector` and stored into the `sortByDate` structure. Those indices were then used to extract the time stamp data from for each event and erroneous indices were removed (lines 523-554). The time data was then stored into the `sortByDate` structure under the given animal number and box in the `latencies` entry (lines 557-565).

PROCESSING

In order to extract higher resolution temporal information, the response latency was calculated for all responses across 15 minute time bins throughout the trial. The number of bins was set to 4 and the bin bounds were calculated assuming a 60 minute session (lines 635-636). Using nested for-loops, empty entries were created in `latencies.analysis` to store incorrect latency and correct latency information for the whole run and within bins (lines 646-653). The `press_latency_vector` and `plb` data was then saved in temporary vectors (lines 655-656). To do the latency calculation, the second press was first determined to be correct or incorrect from the coded values in `press_latency_vector` (lines 667, 680). The latency of each press was then calculated by subtracting the time at the end of the delay from the time of the second lever press using the coded values stored in `press_latency_vector`; this data was then stored in the corresponding entries in `latencies.analysis` and in the appropriate bins (lines 668-678, 681-690).

ANALYSIS

AVERAGE LATENCY, TOTAL RESPONSES, PERCENT CORRECT

The binned latency times were then used to calculate the average, standard deviation, number of

incorrect and correct latencies, total number of responses, and percent correct within each bin; those values were stored in the `bins` entry under `latencies.analysis` (lines 700-721). The averages and standard deviations were calculated using the `mean` and `std` functions. The number of correct and incorrect responses were calculated using the `length` function and these values were added to determine the total number of responses per bin. Finally, the percent correct was calculated by dividing the number of correct alternations by the total number of responses.

AVERAGE ACCROSS TEST GROUPS

To average the latencies, total responses, and percent correct across test groups, vectors containing test days for each group were declared (lines 767-800). A new structured variable named `binPlot` was declared to store the full vectors of the percent correct, total number of responses, correct latencies, and incorrect latencies for each group separated by time bins (lines 819-825). The aforementioned values were then copied from their respective places in `sortByDate` into `binPlot` for each treatment group and test day (lines 827-995). The mean, standard deviation, and standard error was then calculated for each data type, test group, test day, and mean (lines 997-1006). Mean and standard deviation were calculated using `nanmean` and `nanstd` to account for `NaN` values generated by dividing by 0 and standard error was calculated by dividing the standard deviation by the square root of the length of the respective vector (`sqrt(length(…))`). The same was repeated for the morphine test group across all morphine and withdrawal trial days (lines 1026-1073).

COMMUNICATION

RASTER PLOTS

To create a raster plot that fully captured the animal's behavior, information about which lever is being pressed, whether that press was correct or incorrect, and the latency of that press must be represented. In order to do so, black ticks indicate the first press (to give a sense of latency), and the remainder of the information is represented in the second press – the alternation attempts. Correct attempts were indicated by a positive tick, incorrect attempts were indicated by a negative tick. Right lever presses were indicated by a blue tick, left lever presses were indicated by a yellow tick. In order to generate the raster, correct and incorrect alternations on each lever and delay start data (to represent the first lever press) was extracted from each date and animal in the `latencies` entry of `sortByDate` (lines 578-582). First press was plotted as a small black tick centered on the time axis, correct responses on the right lever were plotted as a blue positive tick, incorrect responses on the right lever were plotted as a blue negative tick, and so on (lines 584-589).

LONGITUDINAL PLOTS

In order to visualize how animal behavior changes over multiple sessions, longitudinal plots are used. First, the data was parsed by cohort (ACSF or nor-BNI microinjected animals). The days that each animal in that cohort was run is determined by evaluating which indices of the `percentcorrect` matrix for that animal have a value and storing the respective days in `daysTemp` (lines 196-200). Plot with the given percent correct and total response numbers on each day for each animal is generated (lines 201-209). A cell array containing the respective value for each animal on each day is created (lines 210-211). Those cell arrays are then translated into matrices with `NaN` values filling empty entries (lines 219-223). The average, standard deviation, and standard error within each day across animals is then calculated using `nanmean`, `nanstd`, and dividing the standard deviation vector by the square root of the length of that vector, respectively; the values were stored in new vectors (lines 225-227). A plot of the mean and standard error of the given value across days was then generated and labels were included to mark test days (lines 230-248). This was repeated for both cohorts to visualize both percent correct and total responses across days (lines 251-378). A similar method was repeated to visualize percent correct and total responses across the morphine test days (lines 391-500).

BIN PLOTS

To generate plots that show change between baseline and test days in the average correct and incorrect latencies, percent correct, and total number of alternations within bins, bar graphs were generated. The bar plots were separated by bin and difference colored bars were used to differentiate a baseline day from a test day and incorrect versus correct latencies. For each cohort and test group, the latency, performance, and response data stored in the `binPlot` structure was extracted and stored into a matrix (lines 1160-1193). A bar plot with error bars was then generated for each cohort, test type, and data category (lines 1195-1229). The latency plots were generated with 4 bars per bin: baseline correct, baseline incorrect, test day correct, test day incorrect. The percent correct and total response plots were generated with 2 bars per bin: baseline and test day.

LATENCY HEAT MAPS

To observe the change in latencies within bins across all of the morphine test days, heat maps were generated. The mean and variance of the binned correct and incorrect latency data was sorted and copied into matrices such that the first four rows were the 15-minute bins of animals 1-4, the next four rows were the 30-minute bins of animals 1-4, and so on; the columns represented each morphine test day (lines 1097-1109). Heat maps of the average correct, average incorrect, variance of correct, and variance of incorrect latencies across animals, bins and days were generated (lines 1118-1130).

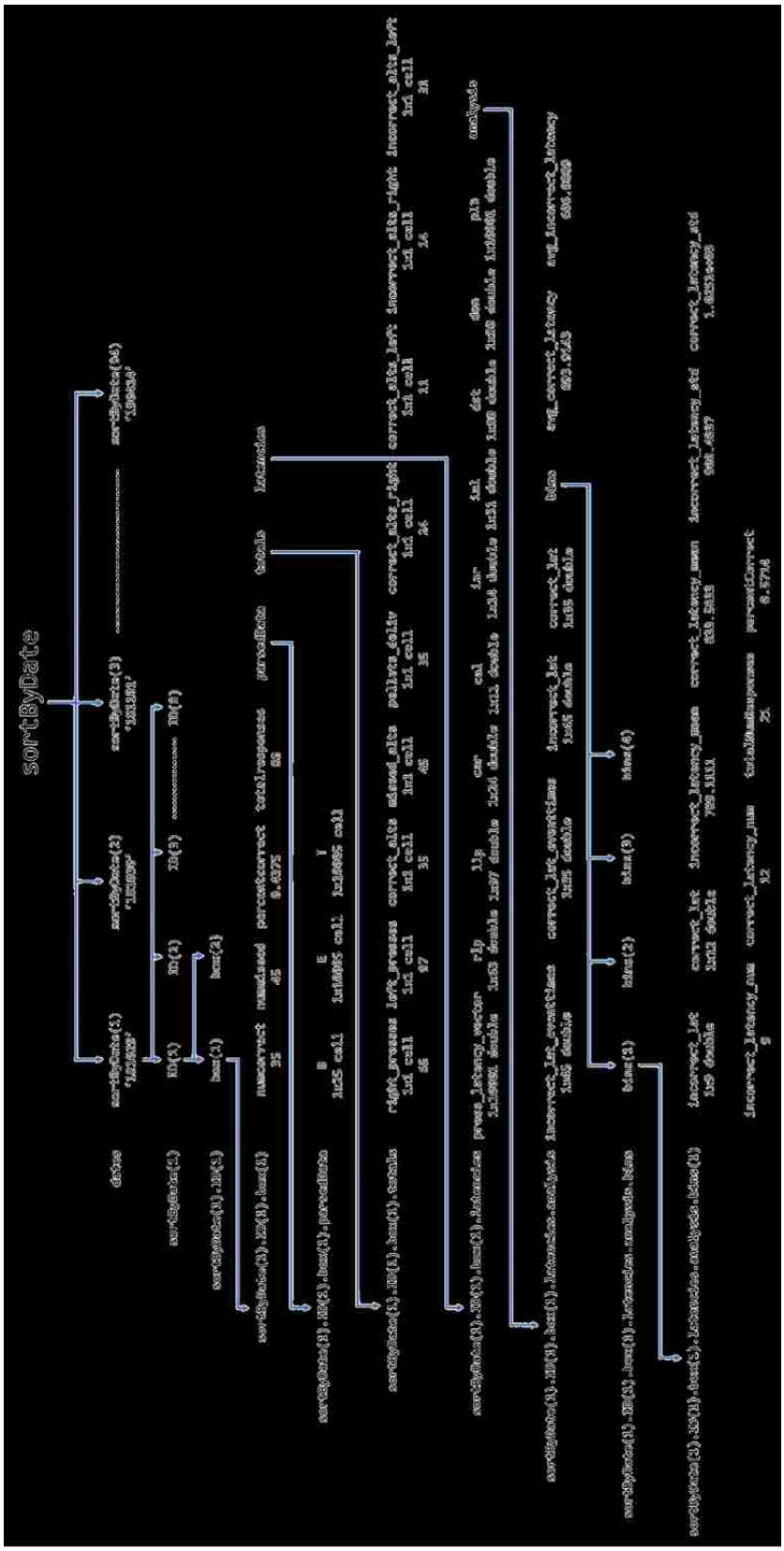

Figure 8: sortByDate Structured Variable Diagram

Diagram depicting the database-like structure of the variable holding the delayed alternation data. Bolded text depicts a data field. Arrows depict the relationship between fields and the data within those fields. The column of bolded entries on the left depicts the variable call used to retrieve the data to the right.

RESULTS

U50,488 AND STRESS

The computational methods described above allowed for efficient and robust handling, processing, analysis, and communication of the delayed alternation data. Classical presentation of delayed alternation data includes percent of correct alternations and perseverance (repetition of the same lever press) for a given trial. However, those summaries lose much of the temporal information of an animal's behavior. To visualize the complete response activity of an animal during a test session, raster plots were used. Representative raster plots of a mouse with a control microinjection of ACSF into the PFC are shown (Figure 9). As the animal progresses through training, their performance (percent correct) increases from 30% on the 2-second-delay paradigm to 74% on the 10-second-delay paradigm over the course of 94 days (Figure 9, plots 1-5). The raster plot presentation conveys information about the animal's preferred lever. From the first plot, it is clear that the animal prefers the left lever. This preference results in perseverance which causes the untrained performance to be worse than chance (50%). On day 95, the animal was administered systemic U50,488 prior to being placed in the operant chamber (Figure 9, plot 6). Administration of U50,488 resulted in a drop in performance back to untrained levels.

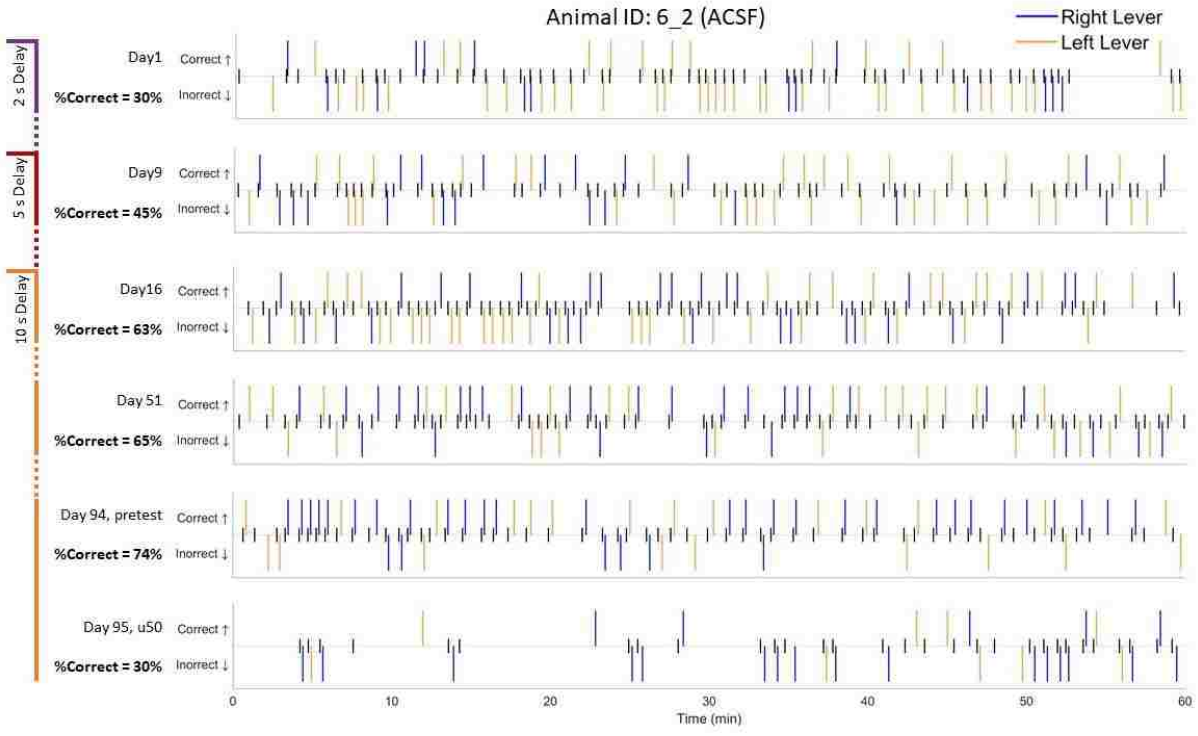

Figure 9: Raster Plots of Delayed Alternation Behavior Across Training and on U50 Test Day

Representative raster plots of a control mouse with micro injection of ACSF into the PFC. Plot 1 shows the first day of alternation training with a 2 second wait period. Plot 2 shows the first day of training with a 5 second wait period. Plots 3-6 are with 10 second waits. Plot 5 (pretest) shows standard performance of a trained animal (65-75% correct). Plot 6 shows reduction of performance on a U50 test day. Plots numbered 1 (top) to 6 (bottom).

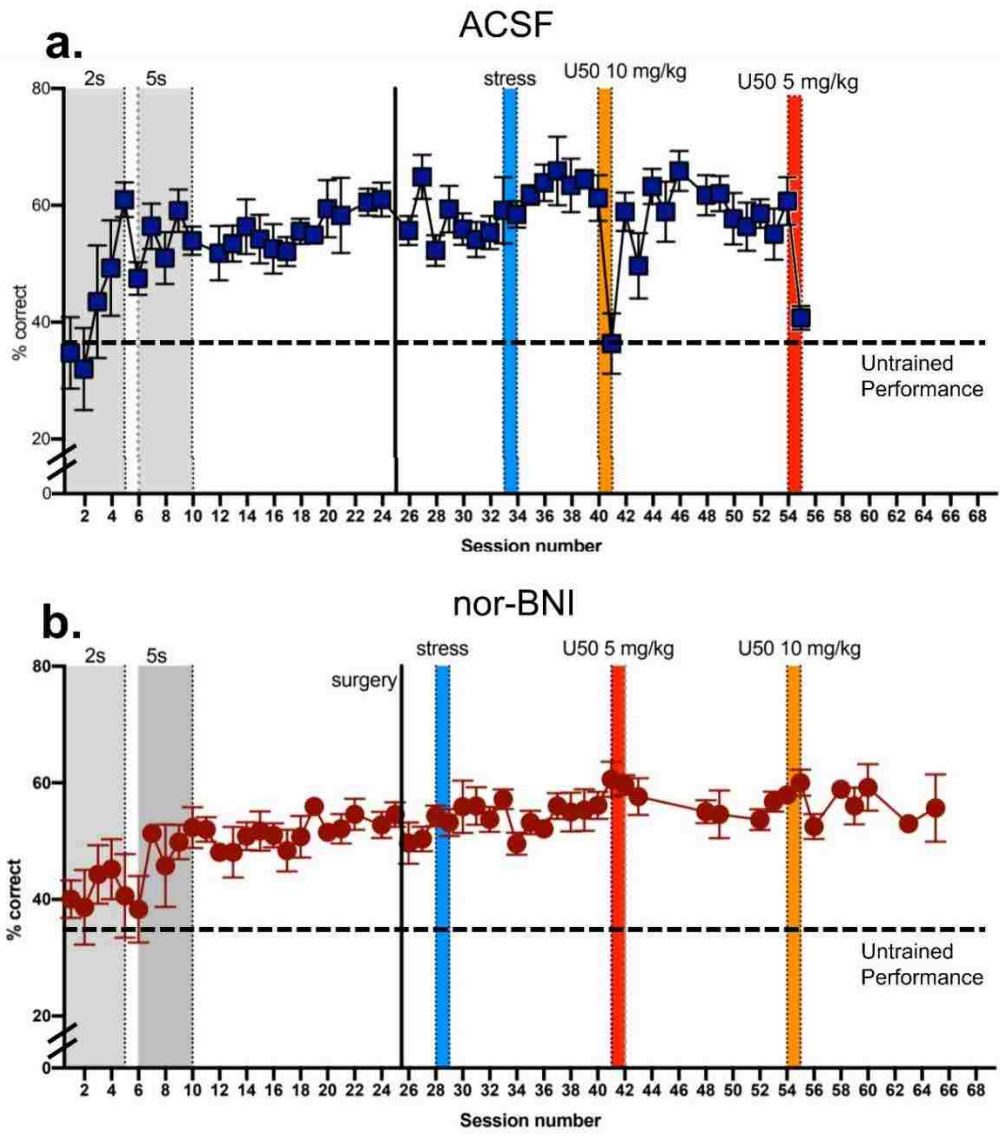

Figure 10: Performance of ACSF and nor-BNI Treated Animals Over Training and Test Days

Longitudinal performance (% correct) of control animals with ACSF microinjections **(a)** and nor-BNI microinjected animals **(b)**. The training period begins with 2s and 5s delay training (gray bars) followed by 10s delay periods for the remainder of the testing. Test days are depicted by colored bars: blue = forced swim stress, red = 5 mg/kg U50 pretreatment, orange = 10 mg/kg U50 pretreatment. **(a)** U50, but not stress, reduces performance to untrained level for control animals. **(b)** This reduction of performance is blocked by a micro-injection of nor-BNI into the PFC.

This training trend and U50 effect was observed across all animals in the ACSF test group (Figure 10.a). U50 doses of 10 mg/kg (orange) and 5 mg/kg (red), but not forced swim stress (blue) reduced performance to untrained levels. This effect was blocked by local nor-BNI in the PFC (Figure 10.b) suggesting that KOR in the PFC is required for this KOR-mediated effect on delayed alternation performance.

The raster plots also convey information about temporal changes in performance within a given trial. For example, during the U50 test day, the animal had a reduction in total number of

responses during the first 30 minutes of the trail (Figure 9, plot 6), likely due the same effects observed in the DRL performance (see Figure 3). The response frequency was recovered by the end of the trial. To parse out whether the change in performance was due to reduced response rate, the behavior factors were analyzed in 15-minute bins (Figure 11). As observed in the raster plots, the total number of responses was reduced during the first 30 minutes after U50 administration (light blue), but the response frequency recovered to control levels (dark blue) by the end of the session (Figure 11.a, bottom plot). Importantly, the reduction in performance after U50 administration was observed through the duration of the session, even after the response frequency was recovered (Figure 11.a, middle plot). This suggests that the U50 mediated performance disruption was not simply due to a reduced response frequency.

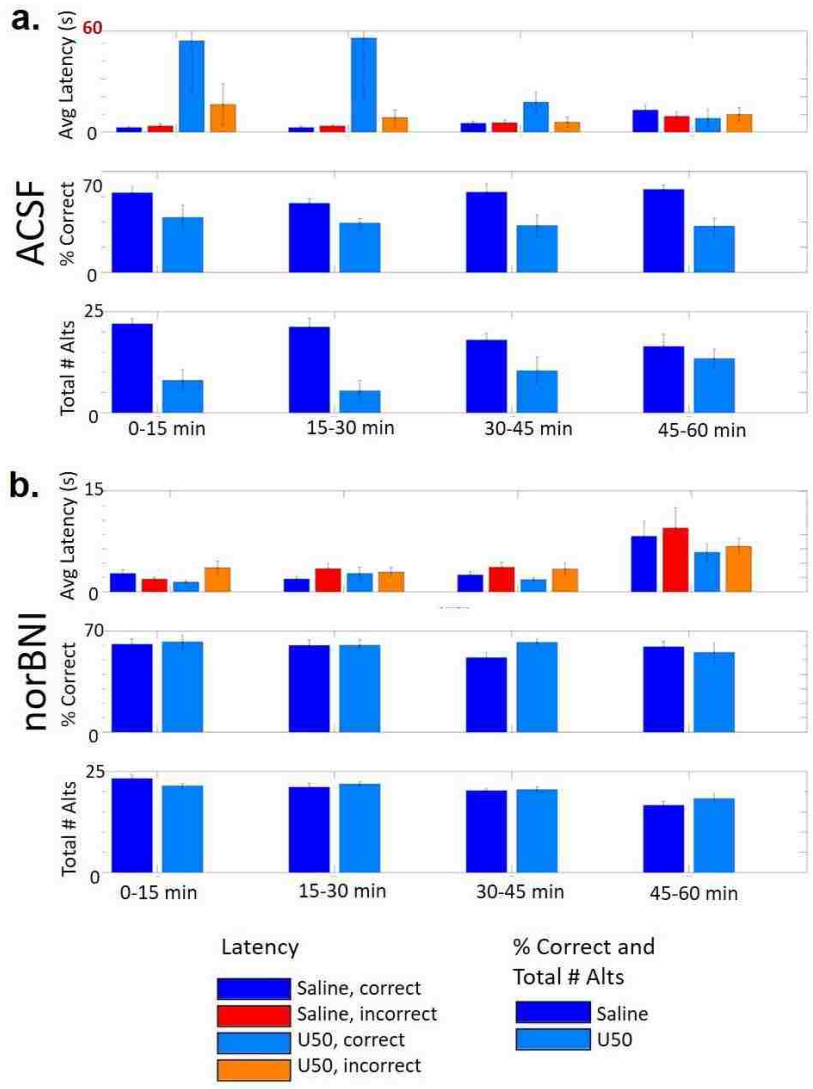

Figure 11: Temporal Effect of U50 on Behavior Within a Session

Average latency (top), percent correct (middle), and total number of alternation attempts (bottom) of ACSF **(a)** and nor-BNI **(b)** microinjected animals on saline and U50 test days. U50 pretreatment decreases the percent correct throughout the duration of the trial, decreases the total number of responses during the first 30 minutes, and increases the latency to respond during the first 30 minutes. All effects are blocked in nor-BNI treated animals.

In addition to the changes in performance, the average latency to make a response during the first 30 minutes after U50 administration was increased compared to baseline (Figure 11.a, top plot, note the change in scale). This increase may be due to locomotor effects; however, the effect appears more prominent on correct alternations (blue) than incorrect alternations (red) suggesting that there may be a cognitive component involved in this timing effect as well.

MORPHINE WITHDRAWAL

To visualize the effect of escalated morphine administration and spontaneous withdrawal on delayed alternation behavior, raster plots were generated (Figure 12). Prior to morphine administration, the animal was responding at a typical trained level (Figure 12, plot 1). On the fifth day of increased morphine administration, the animal had a disrupted response pattern during the last 30 minutes of the session, but their performance was similar to baseline (Figure 12, plot 2). Over the course of a 5-day spontaneous withdrawal period, the animal's performance increased above typical trained levels (Figure 12, plots 3-4).

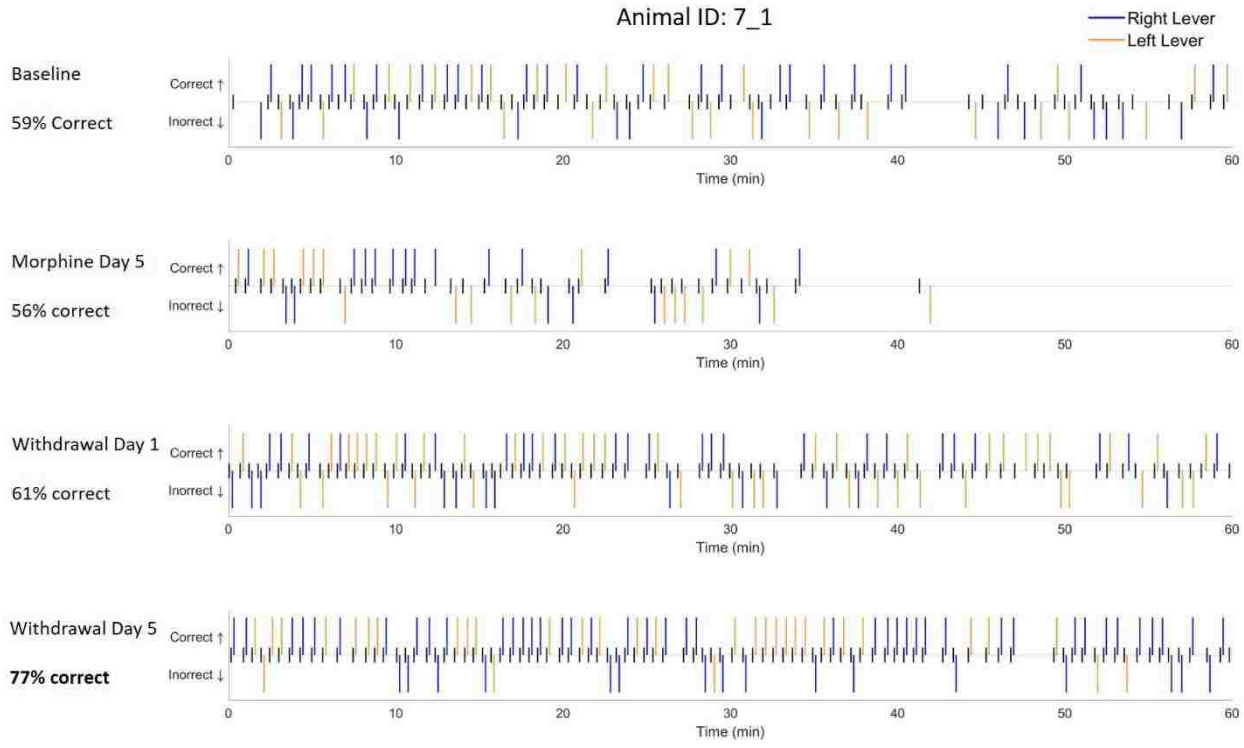

Figure 12: Raster Plots of Behavior During Morphine Treatment and Withdrawal

Representative raster plots of a mouse treated with escalating doses of morphine followed by spontaneous withdrawal. Plot 1 shows standard trained performance. Plot 2 shows the last day of morphine treatment. Plot 3 shows the first day of morphine withdrawal and Plot 4 shows the increase in performance observed on the fifth day of morphine withdrawal. Plots numbered 1 (top) to 4 (bottom).

This surprising result was observed across all animals who went through the morphine treatment paradigm (Figure 13). On the first high dose day of morphine treatment (blue, day 2), the total number of responses dips to about 30 percent of baseline (gray) with recovery over the next 3 days of treatment (Figure 13.b). This dip in response frequency is likely due to the hypolocomotive effects of morphine and the recovery in response frequency is likely due to morphine tolerance (Timár at al., 2005). The delayed alternation performance was not significantly affected by morphine treatment, although the variance of performance between animals increased (Figure 13.a). Interestingly however, spontaneous morphine withdrawal (orange) resulted in an improvement in performance from baseline over five days and a sustained increase in total number of responses.

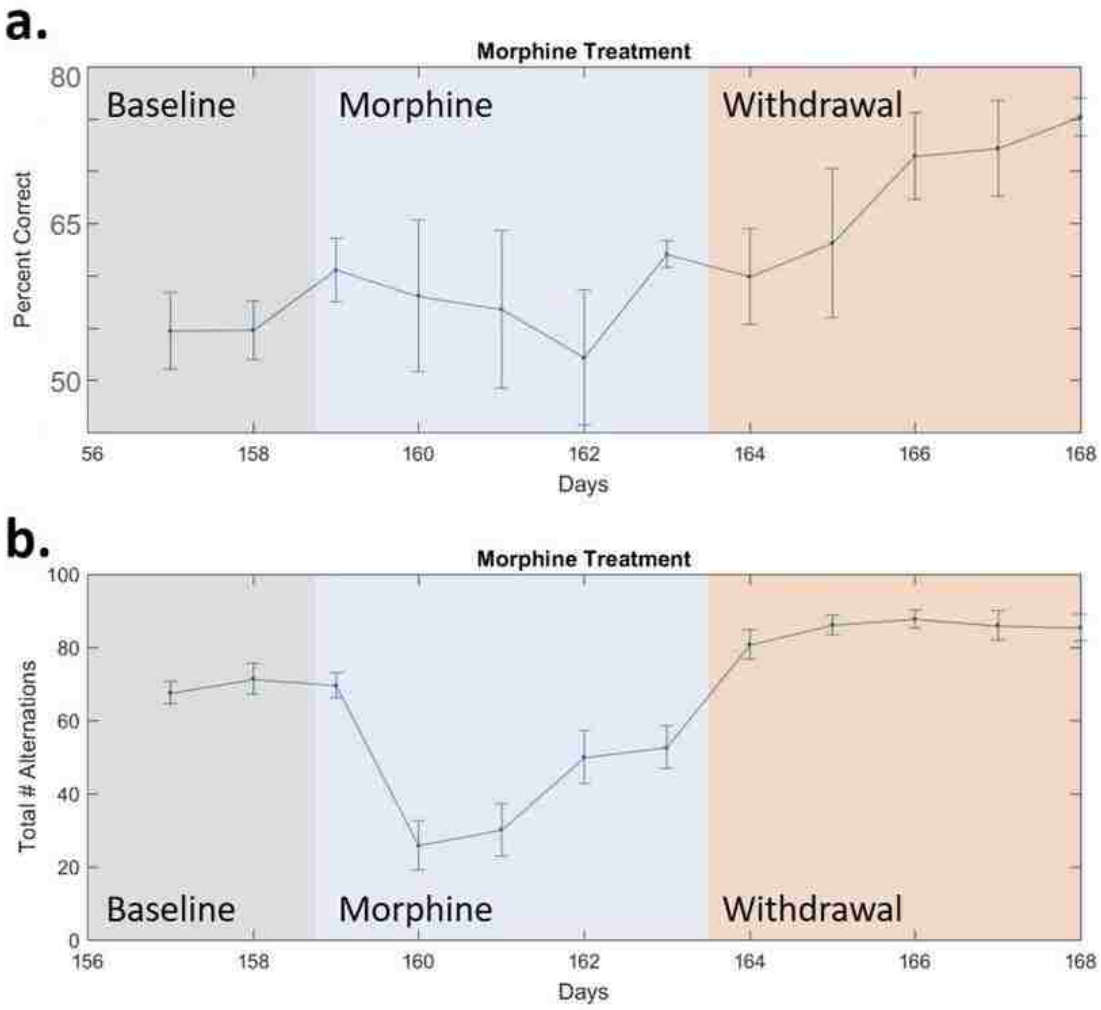

Figure 13: Delayed Alternation Behavior Across Morphine Treatment and Withdrawal

Longitudinal performance (% correct) **(a)** and total number of alternation attempts **(b)** for animals prior to morphine treatment (gray), during morphine escalation (blue), and during spontaneous withdrawal (orange). Morphine treatment initially reduced the total number of responses but had little effect on performance. During the withdrawal period, animals increased their total number of responses and continually increased their performance over the 5 days observed.

Additionally, the variance in the total number of responses during the withdrawal period decreased, and observation of the raster plot suggests that the animals are responding quickly and regularly. To test this observation, the average response latencies and latency variance was calculated for both correct and incorrect responses (Figure 14). The response latency increased during days 2-4 of morphine treatment likely due to the locomotor effects mentioned previously. During morphine withdrawal, both the average response latency and the variance of the latencies decreased suggesting that the animals were responding more rapidly and more consistently. Further investigation into these effects are required to draw a conclusion, but one hypothesis is that the animals were more motivated for a sucrose reward during withdrawal because they had lost body mass during morphine treatment, thus improving their attentions and performance.

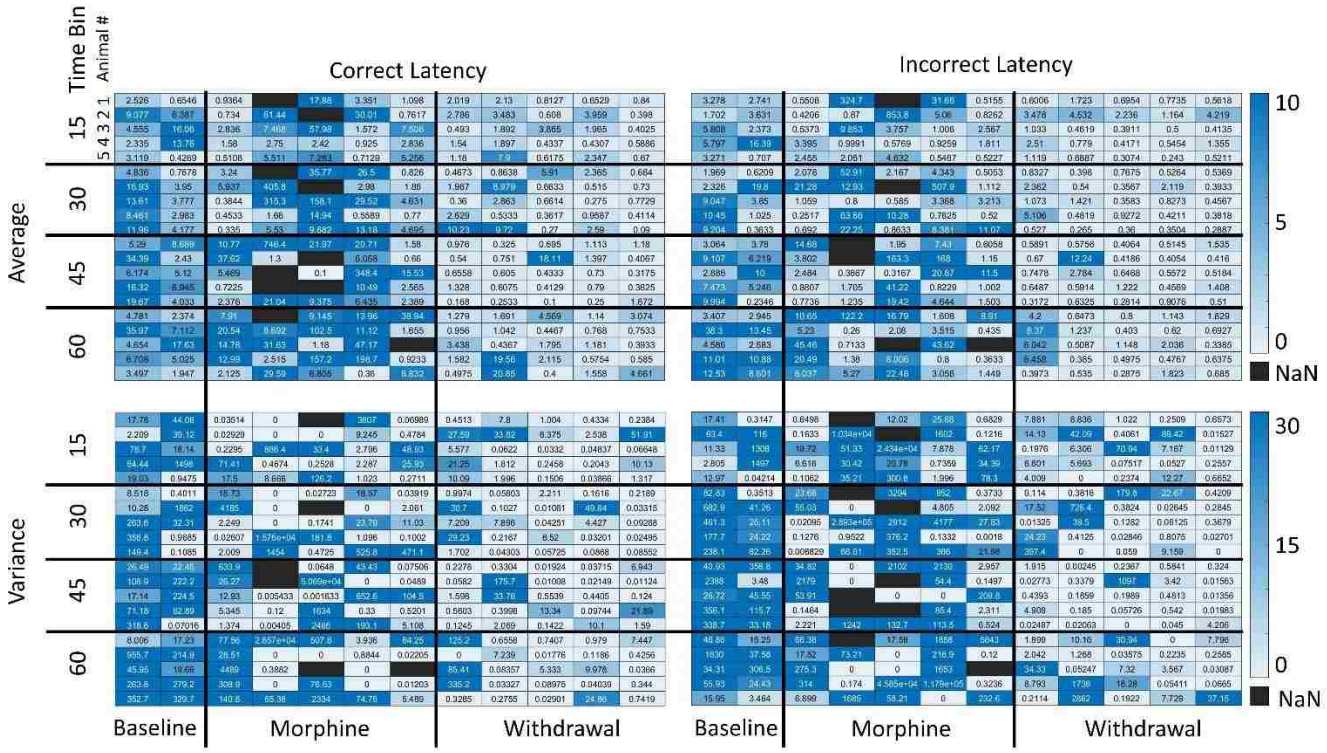

Figure 14: Response Latency During Morphine Treatment and Withdrawal

Heat map showing the average (top) and variance (bottom) of correct (left) and incorrect (right) response latencies during 15-minute bins within test days during baseline, morphine treatment, and withdrawal periods. Morphine withdrawal decreases both the average latency and latency variance for all animals across all time bins.

My findings will be submitted for publication: Abraham AD, Casello SC, Andrews MM, Rivera ZMG, Land BB, Chavkin C. Prefrontocortical kappa opioid receptor activation disrupts working memory via an arrestin-dependent mechanism.

CONCLUSION

The principal findings of the studies presented in this chapter provide insight into the role of the KOR-dynorphin system on two important aspects of motivated behavior: inhibitory control and working memory. The DRL task demonstrated that KOR-activation in the VTA, either by stress or pharmacological means, disrupts behavioral inhibition in an arrestin-independent manner. The operant delayed alternation task demonstrated that pharmacological KOR activation in the PFC disrupts performance in a working memory task, but that rFSS does not cause PFC-KOR dependent disruptions in working memory. Additionally, spontaneous withdrawal from escalated morphine administration enhance delayed alternation performance. Collectively, these results demonstrate that KOR activation causes cognitive disruptions that affect important aspects of motivated behavior, potentially explaining clinically relevant relationships between stress and various cognitive and behavioral disorders.

CHAPTER 3 – CIRCUIT

OVERVIEW

Chapter 3 introduces the importance of neural circuit plasticity on the regulation of behavior in response to internal and external inputs. This information details the short-term and long-term synaptic changes that can occur as a result of adaptive and maladaptive responses to external pressures. Stemming from this context, the effects of stress on the production of both adaptive and maladaptive changes in synaptic activity are described. Finally, the current understanding of the role of KOR on affecting neural transmission is detailed.

The role of KOR on the valuation of reward is discussed, especially KOR's effect on changes in reward valuation following stress. This chapter describes an experimental technique, *in vivo* fiber photometry, that can be used to investigate the role of KOR on stress-induced changes in neural activity in behaving animals. The current gaps in knowledge leading to the research questions being investigated are explained. Then, the experimental and computational methods used to investigate these questions are described in detail, with an emphasis on novel computational approaches to analyzing the data produced.

Finally, the experimental results discovered through these assays are presented within the context of KOR-mediated implications on stress-induced changes in reward signaling.

BACKGROUND AND SIGNIFICANCE

Just as a computational system uses electrical circuits to store and integrate inputs, biological systems use neural circuits to store and process inputs to produce appropriate behaviors. Neural circuits are plastic, ever-changing processing units with the ability to adjust to both external stimuli and internal pressures to promote learning and adaptive behaviors. However, under intense or prolonged pressures, changes in neural circuitry can result in maladaptive behaviors.

Changes in neural function can result from effects at the level of single receptors, to cell excitability and morphology, to synaptic transmission, to large circuit dynamics and can cover timescales from milliseconds to years (Destexhe and Marder, 2004). Neurons communicate with each other through chemical and electrical synapses. An action potential is generated by summing the excitatory and inhibitory inputs from all of the synapsing neurons; when the summation reaches a threshold, an action potential is fired. These inputs are generated in the form of neurotransmitters released by the presynaptic cell that bind to receptors on the post synaptic neuron. The threshold of the neuron is determined by its resting potential, density of receptors, and receptor subtypes (Platkiewicz and Brette, 2010).

The strength of a synapse can be altered by the temporal firing pattern of the presynaptic neuron or by the neuropeptides delivered hormonally or by neuromodulatory neurons (Marder and Thirumalai, 2002). Short-term changes can include depression or facilitation in which successive action potentials decrease or increase respectively. These short term changes are typically due to changes in neurotransmitter release – repeated presynaptic firing causes an increased concentration of neurotransmitter in the synaptic cleft; changes in membrane potential – repeated firing causes a build-up of ions within the post-synaptic cell which can cause depolarization or hyperpolarization; receptor desensitization – repeated activation of a receptor resulting in the receptor becoming inactive; or retrograde signaling – the release of messengers from dendrites that act to inhibit or excite presynaptic terminals (Abbott and Regehr, 2004). However, prolonged or persistent short-term changes can result in long-term changes coined long-term potentiation (LTP) or long-term depression (LTD). LTP and LTD can result from changes in presynaptic transmitter release, postsynaptic receptor expression, and morphological changes at the synapse (Bliss and Collingridge, 1993).

These changes in synaptic strength are required for organisms to process and incorporate information about prior experiences (Holtmaat and Svoboda, 2009; Wilbrecht et al., 2010). However, dysregulation of synaptic plasticity results in maladaptive behaviors and has been implicated in a number of stress-related disorders such as addiction, schizophrenia, anxiety, depression, and PTSD (Russo et al., 2009; Hayashi-Takagi et al., 2010; Liu et al., 2017; Flor and Nees, 2014). Stress has been shown to alter cell morphology, specifically dendritic spine density, which alters the strength of synapses (McEwen, 2000; Wellman, 2001; Vyas et al., 2003). Additionally, stress has been shown to alter the receptor type and density expression on neurons, also causing changes in synaptic strength (Campioni et al., 2009).

Changes in neural activity evoked by environmental factors affect circuit dynamics, resulting in short- and long-term behavioral changes that can be adaptive or maladaptive. To determine the mechanisms of how environmental factors, such as stress, affect behavior, investigation into the underlying circuit dynamics in needed. To elucidate the short-term changes resulting from a given experience, *in vivo* neural recording tools, coupled with appropriate analysis methods are necessary. This chapter introduces a method of *in vivo* neural recording that can be used during behavioral experiments, details the inputs and outputs under investigation, and describes computational methods to better utilize and interpret the data produced.

GCaMP FIBER PHOTOMETRY

MOTIVATION

As introduced in Chapter 1, stress activates the KOR-dynorphin system. KOR activation is known to stimulate G-protein signaling resulting in reduction in neural activity by means of ion channel and neurotransmitter release changes (Bruchas et al., 2010). Additionally, KOR activation results in activation of the p38 signaling pathway which has been shown to cause LTD

(Moult et al., 2008). These synaptic changes are likely involved in the aversive learning and reward signaling disruptions associated with stress and drug abuse.

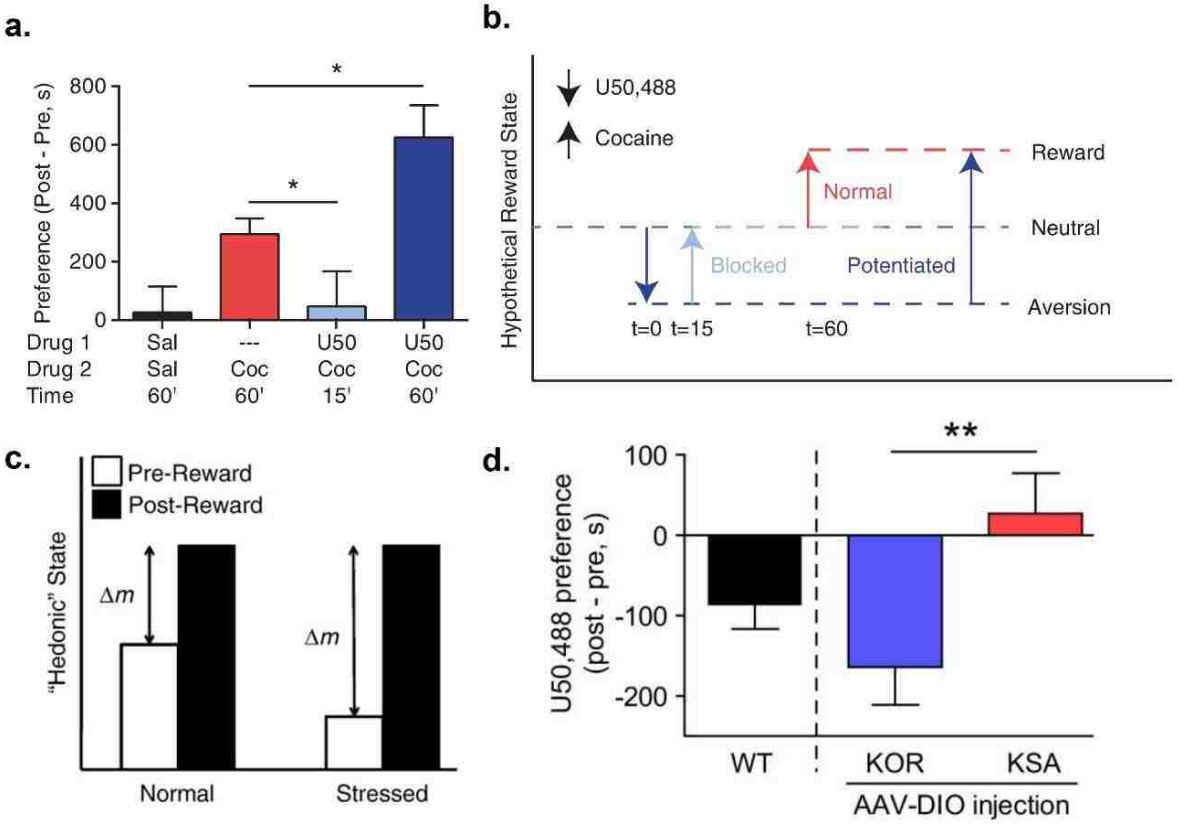

Figure 15: KOR-Mediated Effects on Cocaine Reward Potentiation

(a) Cocaine-conditioned place preference scores for animals receiving cocaine after no pretreatment or after pretreatment with U50,488 either 15 or 60min prior. KOR activation at 15min prior blocks cocaine reward. KOR activation 60min prior potentiates cocaine reward. Error bars represent SEM. Adapted from Ehrich et al., 2014. **(b)** A conceptual model to explain the behavioral results depicted in **a**. Aversion due to KOR agonist treatment within 15 minutes prior to cocaine exposure cancels the rewarding effects of the cocaine, blocking the reward preference. Aversion due to KOR treatment 60 minutes prior to cocaine exposure reduces the reward baseline without blocking the rewarding effects, effectively causing reward potentiation compared to normal cocaine reward. Adapted from Ehrich et al., 2014. **(c)** Activation of the KOR/dynorphin system through stress increases dysphoria, negatively shifting the hedonic or 'mood' state of the animal. This downward shift in baseline creates larger potential positive valence for the drug of abuse, allowing the animal to experience more rewarding effects of the drug. From Bruchas et al., 2010. **(d)** U50,488-conditioned place aversion was restored after injection of DIO-KOR-GFP into the VTA of DATCre/+, KOR knockout mice, suggesting KOR on VTA DAT neurons is sufficient to observe U50-CPA, Adapted from Ehrich et al., 2015.

Studies reporting the effect of KOR activation on reward valence have shown that KOR activation 60 minutes prior to cocaine exposure produces an increase in cocaine reward, however exposure 15 minutes prior blocks the rewarding effects of cocaine (McLaughlin et al., 2006) (Figure 15.a; Ehrich et al., 2014). This effect is hypothesized to be produced by KOR-mediated presynaptic inhibition of dopamine release resulting in a reduction in baseline state (Carrol and Carlezon, 2013; Shippenberg et al., 2001) (Figure 15.b; Ehrich et al., 2014). This effect has been

nominalized as the 'Hedonic State' hypothesis in which a reduced hedonic state caused by stress (or pharmacological KOR activation), would result in a greater change in mood following a rewarding stimulus, effectively increasing reward valance (Figure 15.c; Bruchas et al., 2010).

Additionally, KOR-induced aversion (discussed in Chapter 1), is mediated by KOR activity in VTA dopamine neurons supporting the hypothesis that these behavioral effects are produced by KOR-mediated inhibition of dopamine release in the VTA (Figure 15.d; Ehrich et al., 2015). However, KOR on VTA dopamine (DAT) neurons has not been directly linked to stress induced cocaine potentiation. Nor has the proposed KOR-mediated activity of these neurons been observed during or following stress *in vivo*.

EXPERIMENTAL APPROACH

In order to determine if stress-induced activation of KOR on VTA DAT neurons causes changes in neuronal activity that can explain the increased reward valence of cocaine under stress, tools to allow for recording of neuronal activity in vivo are necessary.

One measure of neural activity is calcium signaling. Calcium ions (Ca^{++}) enter neurons during the firing of an action potential and during synaptic transmission. Therefore, the concentration of Ca^{++} in the cell can be a correlate for neural activity (Yasuda et al., 2004). Genetically encoded calcium indicators (GECIs) such as GCaMP are the state-of-the art for calcium-signaling analysis (Tian et al., 2009). In order to monitor calcium activity during behavior *in vivo*, fiber photometry can be used. Fiber photometry utilizes implanted optical fibers to capture photon release from optically active proteins to detect broad activity of neurons or pathway byproducts within the implanted region (Cui et al., 2013; Gunaydin et al., 2014).

The goal of this assay was to assess the activity changes of VTA DAT neurons during behavioral experiences to inform the role that VTA DAT neurons play in KOR-mediated behavioral changes. Beginning with the fiber implantation, VTA recordings during rewarding experiences such as sucrose pellet interaction followed by recordings during stress paradigms were conducted. Recordings during stress paradigms were done in both wildtype mice as well as mice with genetic excision of KOR from dopamine neurons to determine the regional dynamics and neuronal population effects underlying the encoding of experience in the context of future behaviors.

METHODS

To express GCaMP on DAT neurons in the VTA, an AAV-DIO-GCaMP6m virus was injected into the VTA of DAT-Cre expressing mice. Mice were then implanted with an optic fiber into the VTA and given time to recover. On a test day, mice are connected to the fiberoptic cable and the recording is started. Acquisition is done via the Tucker-Davis Technologies Fiber Photometry System with Synapse software.

While recording, mice were run through two behavioral paradigms: sucrose pellet interaction and repeated forced swim stress (rFSS). During pellet interaction, mice are placed in a housing cage and allowed to freely explore and interact with food pellets placed within the cage. During the rFSS paradigm, mice are placed in water buckets to swim for 6 minutes, then given a 6-minute rest, repeated 4 times in succession.

The software outputs a .tdt file containing the fluorescent values recorded. The .tdt file is converted into a .txt file in a custom program created by Scott Evans.

COMPUTATIONAL APPROACH

The output from this type of recording includes temporal intensity information across the duration of the trial as well as behavioral and pharmacological time points that can be time-locked with the neural signals. Classically, fiber photometry data is down-sampled and assessed for change in fluorescence over a baseline or quantified by the occurrence of transients (spikes in calcium signaling over a threshold). While robust, these analyses only scratch the surface of the available information recorded in each session. In order to better unravel the neural activity and outputs contributing to disruptions in circuit dynamics during behavior, there is a computational need to improve the data handling techniques and breadth of analyses available to researchers.

Specific effort was devoted to exploring various analysis techniques in an attempt to capture a higher resolution of temporal information from the recordings.

HANDLING

The fiber photometry system generates a large amount of data per animal, per run. At a sampling frequency of 1017 Hz, every 16 minutes of recording time generates 1 million data points. With an average rFSS recording lasting 2 hours, each recording contains about 7.5 million data points to be read in and handled.

In order to read each file into MATLAB (version R2017a), a file name is defined (Appendix II, Fiber Photometry Code, Load In, line 10). That filename is then sent to `analyzer` (Appendix II, Fiber Photometry Code, Analyzer Function). `analyzer` concatenates the filename passed with the recording type and file suffix (e.g. `Subject1-190314-120806` becomes `Subject1-190314-120806_greG.txt`) (lines 11). The data is read in as whitespace delimited using `textread` and converted into a vector using `str2double` (lines 13-14). That vector stores each time value followed by the fluorescence signal value, thus every odd entry is a time value and every even entry is a signal value. That vector is then sent into `separate` to divide the vector into two separate vectors containing the time and signal information (Appendix II, Fiber Photometry Code, Separate Function). The `separate` function uses a for-loop to separate the data into time and signal vectors (lines 4-7). The separated vectors are then returned to `analyzer`. `analyzer` then produces a raw data plot to check the recording quality (Analyzer

Function, lines 39-43) and returns the separated data to the main file. The data is then saved as a .mat file under the same name as the original file (Load In, lines 15-16).

PROCESSING

For simplicity, a sample program is included in the appendix (Appendix II, Fiber Photometry Code, Sample Program) to demonstrate the processing, analysis, and communication computations, however this program is a simplified version of the actual programs used to analyze the data.

FILTER

Fiber photometry recordings typically have a low signal to noise ratio. In order to pull out the real signal, a filtering or smoothing function can be used. A typical smoothing function is a smoothing window or moving average. A moving average takes a 'window' of a set size (line 11), averages all of the points within the window (lines 16-17), stores the averaged value in a new vector (line 19), then shifts the window by a set amount and averages again (for-loop iteration). To retain the sampling frequency, the window only shifts by 1 index, but to down sample, the for-loop can be set to iterate by a step greater than 1. To achieve the same effect as a moving window, but without the option to down sample, a simple transfer function can be used (lines 25-27).

NORMALIZE

The fiber photometry system takes photons as an input, then translates photons to electric current to produce a value. Thus, the baseline value depends on the average number of photons entering the system, the resistance along the optical pathway, and the photon capture rate. Thus, the final output value of the system is proportional to but not directly representative of the number of photons being emitted. Therefore, the values outputted by the system must be normalized to a baseline. To normalize, a baseline period must be set (lines 30-31), then those values are averaged to determine an average baseline (line 33). The baseline value is then subtracted from the original signal and the resulting difference is divided by the baseline (line 35) to achieve a $\Delta F/F$ value.

ADJUST FOR DRIFT

Fiber photometry recordings are also susceptible to drift. Drift can occur from the system at the level of the photodetector (Zalocusky et al., 2016) which typically presents as linear drift, or at the biological level from reduction in initial excitability and photobleaching which typically presents as a first order exponential decay. To adjust for these effects, a linear adjustment using

`polyfit` (lines 42-46) or an exponential adjustment using `fit` (lines 49-57) can be performed. Typically, these adjustments are performed on a baseline period and applied to the whole recording (lines 38-39) to ensure that experimental results are not being captured and discarded with the function fit.

ANALYSIS

AVERAGE ACROSS ANIMALS

To observe trends across animals, signals must be averaged with attention to appropriate time-locking of signals. Not shown in the code is time-locking definitions. To average signals across animals, it is important to consistently time-lock events with the signal and compare signals that are recorded during the same behavioral experiences. It is assumed that this has been done to define `swim1_1` (the first swim of animal 1), `swim1_2` (the first swim of animal 2), etc. (lines 60-66). Any signals being compared should be the same length, i.e. `swim1_1` should be the same length as `swim1_2`, etc. Each signal being compared can then be concatenated column-wise into a matrix (e.g. `swim1`) (lines 60-66). To average each signal for the full time period, `mean` is used to average each column of the matrix, i.e. compute the average signal for each animal during the behavioral time period (lines 68-71). To compute an average across animals, a second `mean` is used (lines 68-71). The same logic is used to compute standard deviation with `std`, and standard error by dividing by the square root of the number of animals being averaged (lines 73-76).

FOURIER ANALYSIS

Every time dependent signal can be represented as a weighted sum of sinusoidal signals of constant frequencies. A Fourier transform decomposes a time-domain signal into its respective frequency contributions. The frequency contributions provide insight about the dominant frequencies within a signal which can suggest biologically relevant frequencies and shifts in peaks can inform how a given treatment changes the underlying circuit dynamics.

To transform a time-domain signal into the frequency domain, the discrete fast Fourier transform function, `fft`, is used (line 80). The output of `fft` if a vector, `Y`, the same length, `L`, as the original signal, `sig`, such that:

$$Y(k) = \sum_{l=1}^{L} sig(l) * e^{(-j*2*pi*(k-1)*\frac{l-1}{L})}, 1 \leq k \leq L$$

To reconstruct the frequency contribution, the transform must be divided by the length of the signal which produces a 2-sided frequency spectrum (line 81). To produce a 1-sided spectrum,

only the first `L/2` elements are stored (lines 82-83). To display against the appropriate frequencies, the sampling frequency divided by the length is uses as a scalar multiple on a vector of the length of the original signal (line 84).

COMMUNICATION

AVERAGE SIGNAL PLOT

To display the average signal across animals with the standard deviation of the mean across signals, a column-wise matrix similar to that described in the Average Across Animals section of Analysis is be constructed (lines 93-94). The mean and standard deviation between signals at each time point if computed (lines 99-102). The average signal is then displayed using `plot` (line 105). The standard deviation is displayed using the `fill` / `flip` functions and filled with a transparent color using the `alpha` function (lines 107-109).

BAR PLOT

Using the results computed in the Average Across Animals section of Analysis, a bar plot with error bars can be generated. The average values are plotted as the bar heights (line 115). The error bars are then plotted with the bar centered at the height of each bar and the bar length being defined by the standard error values (lines 116-117).

RESULTS

PROOF OF CONCEPT – FOOD PELLET INTERACTION

The computational methods described above allowed for efficient and robust handling, processing, analysis, and communication of the fiber photometry data. To validate our recording setup and to design the computational methods to be used in future experiments, extended recording trials were done with various pharmacological and behavioral perturbations (Figure 16.a). The four runs contain calcium activity signals from three animals (plots 2,3 are the same animal on separate days) expressing GCaMP on dopamine neurons with the fiber implanted in the VTA (see schematic in Figure 16.b). The perturbations include U50,488 administration, cocaine administration, nor-BNI administration, interactions with food pellets, and opening of the experiment room door.

To determine the consistency of VTA DAT neuron responses to a rewarding stimulus across animals, every instance of a food pellet interaction was extracted and averaged (Figure 16.b). The amplitude and time course of the response was consistent across the 6 instances of food pellet interactions. The calcium signal increase within 5 seconds approaching the food pellet.

The average peak response was a change in fluorescence over baseline of 7.4±2.6 percent at 22 seconds after the initial interaction with the signal returning to baseline 120 seconds after the initial interaction.

These results provide a proof-of-concept validation that VTA-DAT neurons increase activity in response to a rewarding stimulus and that that signal can be recorded, extracted, and analyzed for consistency across animals.

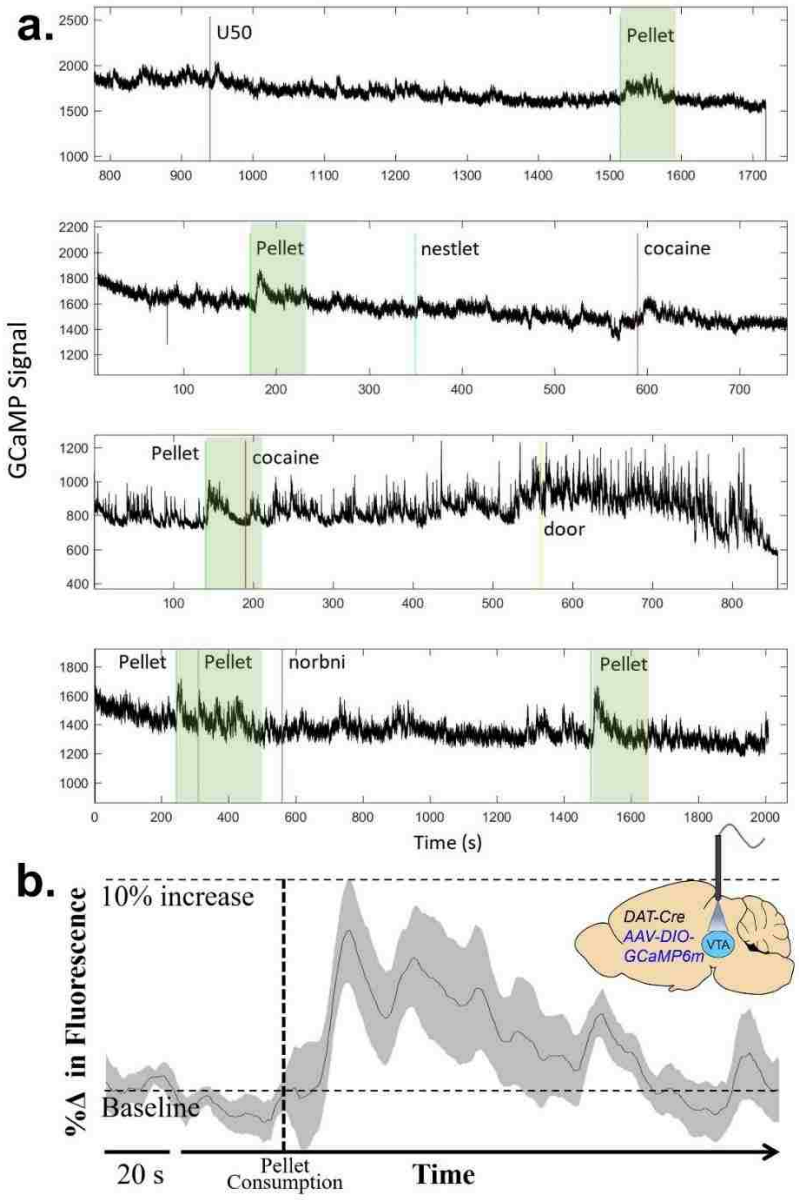

Figure 16: Calcium Activity in VTA Dopamine Neurons During Food Pellet Interaction

(a) Four GCaMP fiber photometry traces from VTA DAT neurons from three animals (plots 2,3 are the same animal) during various perturbations. Green boxes indicate response following interaction with a food pellet. **(b)** Average response to food pellet interaction averaged across 6 interactions. Interaction increased VTA-DAT calcium activity within 5 seconds with activity returning to baseline 120 seconds after the initial interaction.

U50,488 ADMINISTRATION

After confirming that experimental events can be time-locked with a validated fiber photometry signal, the effect of U50,488 administration on VTA DAT neuron activity was investigated. Four animals were injected with U50,488 during fiber photometry recording sessions over the course of three days. Additionally, 3 animals were injected with saline during recording sessions to act as controls. Sample raw traces of a U50,488-injected (day 1) and saline-injected animal are show inn Figure 17.a-b, respectively (pink line marks injection time).

The recordings were smoothed with a moving average function to remove noise and averaged together (Figure 17.c). The recordings had a stereotyped exponential decay drift likely due to photobleaching (Mao et al., 2008). This drift was accounted for by fitting a first-order exponential to the baseline signal (Figure 17.d) and subtracting the fitted function from the signal. After removing the exponential drift, the averaged signal for the U50,488 injected animals is displayed in Figure 17.e. The same method was applied to the control signals, and a larger smoothing window was ran across the signals. The average U50,488 response (red) was plotted against the control signals (gray) (Figure 17.f).

Unexpectedly, U50 appeared to increase VTA DAT activity post injection. To better understand the temporal effects of this treatment, the analysis was separated by treatment day (Figure 17.g). On Day 1, the calcium signal decreased directly following U50 administration and remained suppressed for approximately 15 minutes and didn't reach peak fluorescence until approximately 25 minutes post-treatment. The reverse trend was observed on Day 2 in which U50 administration immediately increase calcium signaling and declined to baseline 25 minutes post-treatment. On Day 3, there was initial suppression of signal followed by a peak in fluorescence 25 minutes post administration. All signals returned to baseline by 50 minutes post-treatment.

While additional investigation into this U50 effect is needed, the initial reduction of calcium signal on Day 1 is consistent with the evidence that U50,488 administration decreases DAT activity (Ehrich et al., 2015). The inhibitory time-course of the signal is also consistent with the time-course of hypolocomotion observed during behavioral assays as demonstrated in Chapter 2 and quantified by Paris et al., 2011. The latter temporal response properties on Day 1 could be indicative of receptor desensitization and/or internalization as characterized by Li et al., 1999. The increase in calcium signal observed on all three days may also be due to compensatory effects within the VTA or within regions projecting to the VTA. While this increase in calcium signaling following U50,488 administration requires additional investigation, it may suggest a post-inhibitory rebound effect caused by the initial suppression of VTA DAT activity.

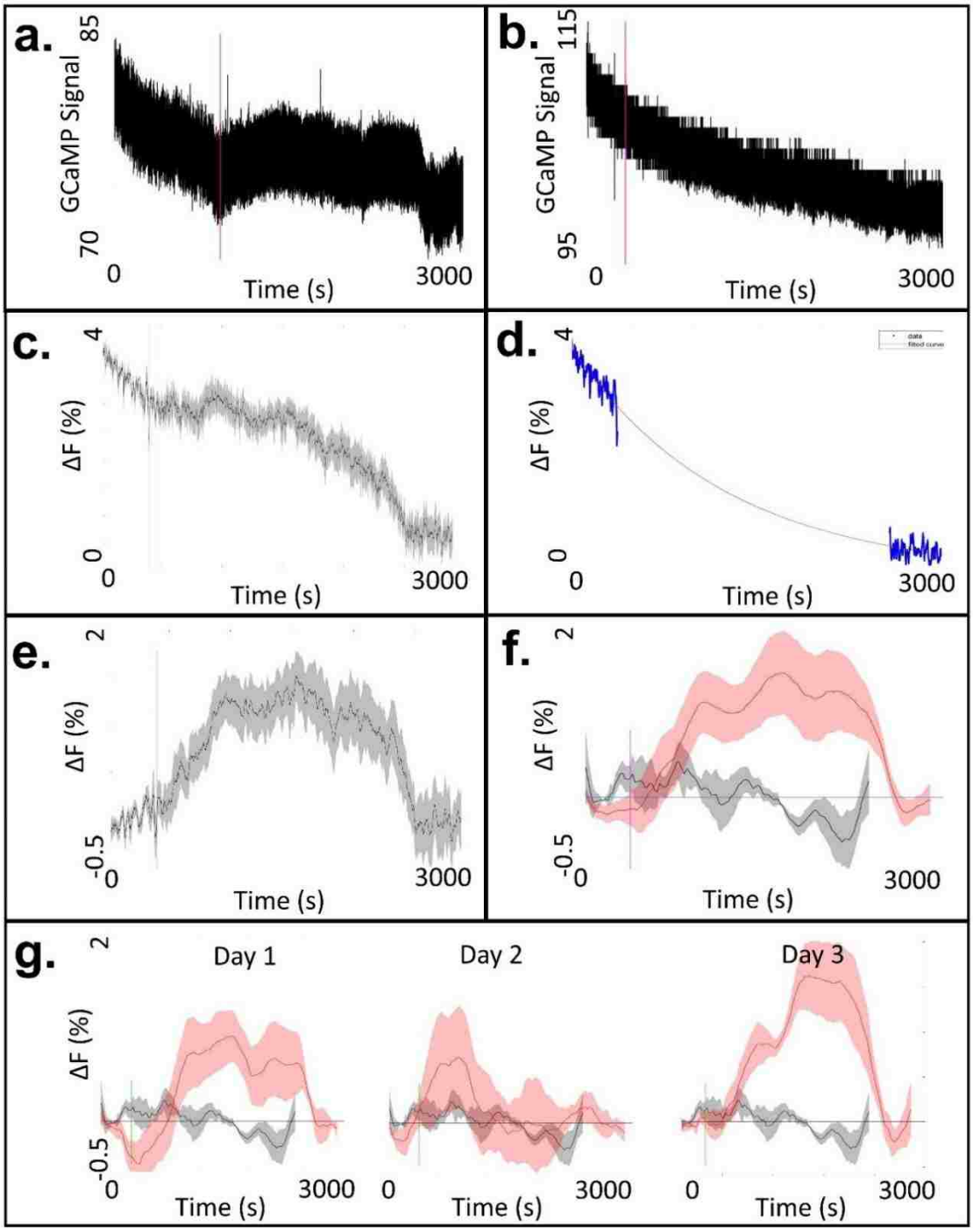

Figure 17: GCaMP Signal Processing and VTA DAT Activity After U50 Administration

Sample raw fiber photometry traces of a U50,488-injected (day 1) **(a)** and saline-injected **(b)** animal. Pink line marks injection time. **(c)** U50 traces after being smoothed with a moving average function to remove noise and averaged together. **(d)** Exponential decay fit line used to adjust the recordings for drift. **(e)** Averaged U50 signals after removing the exponential drift. **(f)** Average adjusted recordings from U50 treated animals (red) and control animals (gray) across all days. **(g)** U50 and control recordings split up by day. Generally, U50 treatment causes an initial reduction in calcium signal followed by an increase in signal.

REPEATED FORCED SWIM STRESS

In order to observe the effects of stress-induced activation of KOR on VTA DAT neuron activity, fiber photometry data was recorded during a repeated forced swim stress (rFSS) paradigm. During this paradigm, animals swim for 4 repeated 6 minute sessions (S1-4, Figure 18) with a 6 minute rest between swims (P1-4, Figure 18). Filtered traces from a representative animal with wildtype expression of KOR (blue) and an animal with conditional knockout of KOR from DAT neurons (yellow) are shown in Figure 18 with the swim periods highlighted.

In animals with normal expression of KOR, an increase in calcium activity following each swim period is observed. Additionally, this increase in activity is potentiated with each swim. This increase in calcium activity following stress is consistent with the increase in calcium activity following U50 administration, however the close time-locking with the behavioral events suggests that this may be indicative of a reward or 'relief' signal following the swim stress as opposed to a compensatory or pharmacokinetic effect. Both the post-swim increase in calcium activity and the potentiation of that increase is suppressed in the DAT flox KOR mice supporting the hypothesis that KOR is directly mediating the stress-induced changes in reward signaling.

This increase in calcium activity following stress may begin to explain the neural mechanism underlying KOR-mediated stress-induced cocaine reward potentiation. Calcium activity in VTA DAT neurons is consistent with reward signaling. A KOR-dependent post-stress increase in calcium activity in that neural population could explain the reward potentiating effects of stress on drugs of abuse.

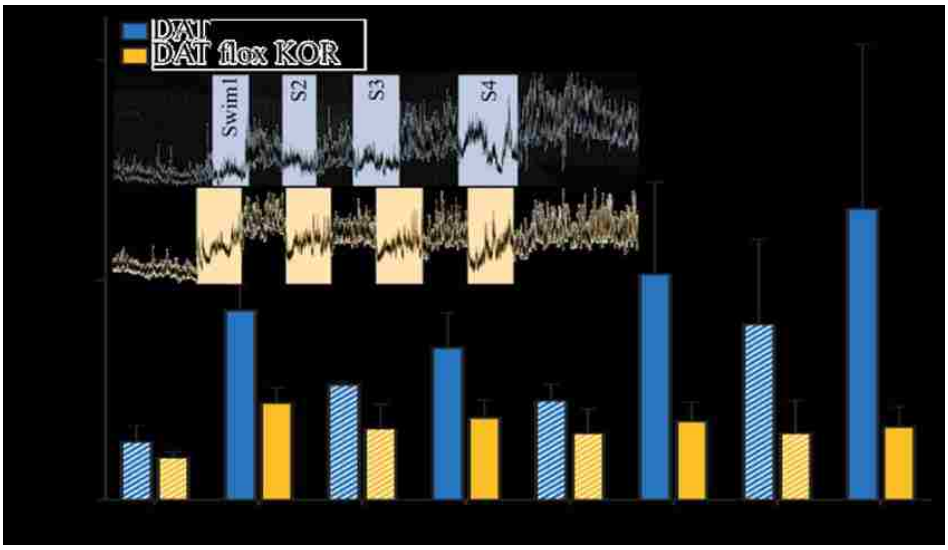

Figure 18: VTA DAT Activity During Repeated Forced Swim Stress

Repeated swim stress potentiates calcium activity in dopamine neurons. S1-S4 show average calcium activity during repeated forced swim stress, and P1-P4 shows average calcium activity in the period immediately following each swim stress. Deletion of KOR from dopamine neurons appears to suppress the increases in calcium activity following swim stress. *p<0.05; Error bars indicate SEM.

CONCLUSION

The principal findings of the studies presented in this chapter provide insight into the role of the KOR-dynorphin system on important aspects of neural activity and circuit dynamics. The *in vivo* GCaMP fiber photometry recordings demonstrated that pharmacological KOR activation causes an initial inhibition in VTA DAT neuron activity followed an increase in activity, potentially suggesting a post-inhibitory rebound effect caused by the initial KOR-dependent suppression of VTA DAT activity. However, the magnitude and time course of this effect changes over successive days of administration and requires further investigation to understand the full implications of these findings. Additionally, rFSS demonstrated a KOR-dependent post-stress increase in calcium activity in VTA DAT neurons. Collectively, these results demonstrate that KOR activation in VTA DAT neurons, either through stress or pharmacological means, cause changes in neuronal activity that could explain stress-induced changes in reward signaling including the reward potentiating effects of stress on drugs of abuse.

CHAPTER 4 – Cellular

OVERVIEW

Chapter 4 introduces the importance of cellular pathways on the regulation of cellular responses to extracellular inputs. This information details the signaling cascades that occur as a result of receptor activation and discusses their implications on cell functioning. Stemming from this context, the signaling cascades resulting from KOR activation are discussed, including their implications on higher level cognitive and physiological changes. Finally, the outputs of the described signaling pathways are discussed, including their implications on receptor inactivation.

This chapter describes an experimental technique, real-time cellular microscopy, that can be used to investigate the role of KOR agonism on an identified cellular pathway. The current gaps in knowledge leading to the research questions being investigated are explained. Then, the experimental and computational methods used to investigate these questions are described in detail, with an emphasis on novel computational approaches to analyzing the data produced.

Finally, the experimental results discovered through these assays are presented within the context of KOR agonist effects on a receptor-inactivating signaling cascade.

BACKGROUND AND SIGNIFICANCE

Just as a computational system uses nodes to integrate inputs to generate a binary output, neural systems use cells to integrate information to produce a similarly binary output. However, unlike computational systems – where the integration is done in a linear manor with a single output, the cells of biological systems have complex, diverging, nonlinear pathways with the ability to integrate, compute, store, and respond to a complex series of inputs to generate a collection of outputs, including but not limited to the binary action potential discussed in Chapter 3. Understanding the cellular mechanisms and pathways underlying adaptive changes in circuit dynamics is necessary to complete the story of how a series of inputs can generate lasting changes in behavior.

Cells receive external inputs via receptors located on the cell surface or within the cell. When a change in the environment activates a receptor, the receptor undergoes a conformational change – a change in the physical structure of the protein – resulting in a cascade of intracellular events. The general model of signal transduction includes a chain reaction initiated by ligand binding to the receptor, followed by a conformational change, activation of enzymes and second messengers to amplify the signal, resulting in transcriptional and nontranscriptional effects (Wu, 2013). This receptor triggered signal transduction influences nearly every physiological output in a biological system.

A common signal transduction mechanism conserved evolutionarily to sense many extracellular signals begins with G protein-coupled receptors (GPCRs) (Bockaert and Pin, 1999). Ligand binding to a GPCR results in recruitment of heterotrimeric G proteins (Gα and Gβγ) and stimulation of guanine nucleotide (GDP to GTP) exchange (Granier and Kobilka, 2012). The GTP-bound α subunits (Gα) dissociate from the β and γ subunits (Gβγ), each effecting target proteins for production of second messengers.

In addition to the second messenger cascades initiated directly by the G protein subunits, GPCRs can initiate signaling cascades through another class of proteins, GPCR kinases (GRKs). GRK pathways are also remarkably conserved throughout biological signaling, however they play a very different role. Once activated, GRKs phosphorylate the active receptor resulting in high affinity binding of arrestin, blocking future G protein binding and activation, resulting in receptor desensitization (Pitcher at al., 1998). While GRKs act as a negative feedback control to prevent overstimulation of target cells, consequences of their regulation contribute to a number of disease classes including opiate addiction, hypertension, and chronic heart failure.

Activation of a single receptor can result in multiple different signaling cascades resulting in an array of variable cellular outputs and physiological implications that can be adaptive or maladaptive. Understanding the mechanisms and outputs of those various pathways is necessary to understand the complex cognitive and behavioral implications of a given environmental stimulus. To investigate a given cellular pathway, the ability to isolate that pathway, perturb it's inputs, and record the given outputs in live cells, coupled with appropriate analysis methods is necessary. This chapter introduces a method of *in vitro* live cell recording that can be used during a series of cellular perturbations, details the inputs and outputs under investigation, and describes computational methods to better utilize and interpret the data produced.

SINGLE CELL HYPERRED IMAGING

MOTIVATION

As introduced, KOR activation results in both a Gβγ, arrestin-independent signaling cascade, as well as a GRK3, arrestin-dependent cascade (Bruchas and Chavkin, 2010). Therapeutically, functionally selective agonists that only activate the Gβγ signaling produce analgesia without the dysphoric effects of unbiased activation (Ehrich et al., 2015; Brust et al., 2016; Schattauer et al., 2017a). However, KOR antagonists can also promote stress resilience including reduction of the dysphoric, anxiogenic and pro-addictive effects of stress exposure (McLaughlin et al., 2003a; Mague et al., 2003; McLaughlin et al., 2006; Carlezon et al., 2006; Knoll and Carlezon, 2010). Thus, both KOR antagonists and KOR agonists have been advancing in clinical trials (Carroll and Carlezon, 2013; Chavkin and Martinez, 2015). However, the functionally distinct signaling responses of KOR activation underlying these conflicting therapeutic benefits are not fully understood.

In addition to the conventional GPCR and GRK pathways, KOR activation initiates other protein kinase cascades. KOR activation of the mitogen-activated protein kinase (MAPK) ERK1/2 occurs in an arrestin-independent early phase and an arrestin dependent late phase (McLennan at al., 2008; Schattauer et al., 2012). However, KOR activation of p38 MAPK occurs only in the arrestin dependent late phase (Bruchas at al., 2006). Additionally, both KOR agonsist and antagonist binding activate a third class of MAPK, c-Jus N-terminal kinase (JNK) (Bruchas et al., 2007; Melief et al., 2011; Schattauer et al., 2017b). Activation of pJNK by nor-BNI ultimately results in the production of reactive oxygen species (ROS) (Schattauer et al, 2017b). This ROS production causes permanent inactivation of the KOR-Gαi/o complex via stimulation of peroxiredoin 6 (PRDX6), requiring new receptor synthesis to restore function.

This suggests that receptor-inactivating KOR antagonists such as nor-BNI are in fact JNK-biased agonists that cause long term effects through this ROS-mediated inactivation. Interestingly, KOR agonists also increase pJNK. However, the signaling mechanisms of this process are unclear, and the physiological consequences of this agonist activation are unknown (Kam et al., 2004; Bruchas et al., 2007).

EXPERIMENTAL APPROACH

In order to investigate the mechanisms and effects of KOR agonist induced JNK signaling, there is a need to elucidate the effects of early-phase, arrestin-independent signaling from late-phase, arrestin-dependent signaling on the production of the functional byproduct, ROS.

In order to monitor the temporal components of cellular mechanisms such as ROS production, real-time microscopy can be coupled with optically active proteins like those used in fiber photometry. Real-time cellular microscopy records a series of images over the course of an experimental time period to track changes over time and allows for the administration of pharmacological agents to observe the cellular effects in real-time. To monitor ROS production, the fluorescent sensor HyPerRed can be used (Ermakove et al., 2014). HyPerRed is expressed by transfection, thus it is a viable readout for ROS production in live cells.

The goal of recording the fluorescence of HyPerRed transfected cells using real-time cellular microscopy was to investigate the time-course of ROS production by KOR agonists under various pharmacological conditions. Beginning with baseline recordings, images were taken of cells after application of U50,488, as well as blocking and reversal trials with naloxone. Monitoring of ROS production after application of KOR-targeted pharmacological agents helped to determine the cellular mechanisms underlying the physiological response to stress-induced dynorphin release.

COMPUTATIONAL APPROACH

The output from real-time cellular microscopy includes a series of images with pixel intensity information captured over time. Classical analysis of cellular microscopy requires selection and numbering of cells by hand prior to computer-facilitated quantification. This process is time consuming and leaves room for biasing and arbitration of results (Ninomiya et al, 2016). In order to efficiently and appropriately analyze real-time cellular microscopy images, there is a computational need to automate the cellular identification, tracking, and quantification steps.

Specific effort was devoted to efficiently, accurately, and robustly identifying cells from the background in order to automate the quantification and tracking of their fluorescence over time.

HANDLING

Live cell imaging generates a large amount of data per imaging session. Each image contains 262,141 pixels. With 3 channels per position, 3 color values per channel (red, green, and blue), approximately 60 images per position, and four positions per session, there are over 500 million point values generated per session.

The imaging system exports the image series as .lif files. ImageJ is used to load the .lif files and convert the image series into GIFs. The GIFs contain a series of 3 images for each time point, one for each channel. To separate the channels, a file path is defined in MATLAB (version R2017a) and every file within that path is loaded into from the working directory (Appendix II, Live Cell Imaging Code, Cell Quantifier, lines 4-10). The channels containing HyPerRed frames and cell membrane frames are defined in the self-defined function `loadFrames` (Quantification Functions, lines 3, 6). The frames of interest from the .gif files are then imported using the `imread` built-in function (lines 4, 7). The frames are saved into 4-dimensional matrices containing the red, green, and blue intensities of each pixel for each frame (e.g. `imgHyper(1,2,3,4)` would be the pixel in the first row, of the second column, the blue pixel intensity (red = 1, green = 2, blue = 3), of the 4th frame in the series) (Cell Quantifier, line 18).

PROCESSING

IDENTIFY CELLS FROM BACKGROUND

For each frame, the red pixel values of the HyPerRed channel are sent to the `findClusters` function (Cell Quantifier, lines 25-27). The pixels over a given noise threshold value are amplified by a given enhancement amount (Quantification Functions, lines 15-23). Noisy dark pixels are filled using `imfill` and a local Laplace filter with a sigma of 0.1, alpha of 5, and with 10 levels is run across the image to smooth sharp changes between values, i.e. reduce pixel noise (lines 26-31). An opening function using a disk shaped structured element of radius 3 is

run across the image to remove noisy pixel groups and protrusions smaller than a 3 pixel radius (lines 33-34).

A self-defined cell border finding algorithm was created that, for every pixel in the image, evaluates the pixels above and below and to the left and right a set distance, `dist`, away from the center pixel (lines 45-51). If the pixel above (or to the left) is greater than a given upper threshold, and the pixel below (or to the right) is less than a given lower threshold (or vice versa), the center pixel is determined to be on a border and it's index is saved in a `clusterBorders` matrix (lines 53-63). The fully evaluated border matrix is then filtered with closing (dilation followed by erosion), pixel bridging, diagonal filling, and thinning functions to connect close unconnected pixels and then thin the borders to a minimal thickness (lines 65-68).

The defined cluster borders are then sent to the `fillCluster` function (Cell Quantifier, line 31). Within `fillCluster`, an algorithm was defined to evaluate whether a cell border coincided with the edge of the image and to fill the border if so (Quantification Functions, lines 73-100). Then, the defined cell regions are filled with the `imfill` function (line 102).

The defined filled clusters are then sent to the `watershedClusters` function to separate overlapping cells within a cluster (Cell Quantifier, line 34). Within `watershedClusters`, the black and white clusters image is inverted and a Euclidean distance transform, `bwdist` is ran across the image to determine the distance between each pixel and the nearest nonzero pixel (Quantification Functions, lines 109-111). All minima in the output of the Euclidian distance transform within the boundaries of the inverted cluster matrix with a value less than 3 (i.e. any pixels whose closest non-zero pixel is less than 3 pixels away) are then stored in a new matrix (lines 112-113). This matrix is then passed through the `watershed` function to separate cells whose borders were overlapping other cells by a radius of 3 pixels (line 114). All watershed cells are set to pixel values of 1 and the image is converted to a logical matrix (lines 116-128).

The watershed clusters are then sent to the `labelClusters` function to filter cells based on size and label the passing cells (Cell Quantifier, line 37). Within `labelClusters`, the filled, watershed cells are numerically labeled using `bwlabel` (Quantification Functions, line 135). The labeled cells are then evaluated to determine if their area is within a defined range for reasonable cell sizes as chosen by visual assessment (lines 137-138) and identified cells smaller or larger than that range are deleted from the image (lines 140-151). The cell size filtered image is then relabeled (line 153). A new border map is then created using the newly defined cell image with the same algorithm used to define borders originally (lines 156-199).

ANALYSIS

CELL TRACKING

To identify and track the same cells across all frames of a given recording, the labelled cells were sent to the `getCentroids` function (Cell Quantifier, line 40). Within `getCentroids`, the `regionprops` function is used to calculate the centroids (center of mass) of every cell region in

the labeled image (Quantification Functions, line 205). The x and y values of the centroid positions are then stored in a matrix (lines 207-211).

The centroid positions for all cells across all frames of a given recording are then sent to the `defineCells` function (Cell Quantifier, line 48). Within `defineCells`, the x and y positions of the centroids for each frame are stored into matrices (Quantification Functions, lines 251-265). The centroids from each frame are then clustered to determine which centroids are most likely to belong to the same cell across frames (lines 273-275). Centroids that do not appear in at least one third of the frames for a given centroid cluster are deleted and numbers are assigned to each centroid to denote its given cell ID (lines 273-309).

The cell numbers for all cells across all frames of a given recording along with the cell labeled images are then sent to the `labelCells` function (Cell Quantifier, line 51). Within `labelCells`, the cell labeled image is relabeled with the cell numbers defined by the centroid cluster analysis (Quantification Functions, lines 316-334). A new border mask for the cell labeled image is created using the same algorithm described previously (lines 337-366).

CELL QUANTIFICATION

To quantify the fluorescence of each cell, the cell label mask and original HyPerRed image are sent to the `quantifyCells` function (Cell Quantifier, line 58). Within `quantifyCells`, the cell label mask is converted into a 3-dimensional matrix comprising a stack of binary masks for each individual cell (i.e. `cellMask(:,:,1)` is a cell mask for the first cell in the imaged position) (Quantification Functions, lines 378-389). Each cell is then quantified by multiplying the given cell mask by the original HyPerRed image (line 397). The size of the cell is calculated by summing across the cell mask and the total fluorescence is calculated by summing across the product of the cell mask and original image (lines 399-400). The average fluorescence of the cell is calculated by dividing the total fluorescence of the cell by the cell size (line 402).

To quantify full frames, sums across the original HyPerRed images are taken (Cell Analysis, lines 47-49). To account for frame-to-frame noise fluctuations in the time-course data, a two-frame-window moving average was used (lines 51-59).

To calculate the average fluorescence across a treatment group for the full time series, the ΔF/F values are first calculated by averaging the fluorescent values across a set of baseline frames, then dividing all of the frame fluorescent values by that baseline average (lines 90-92). The ΔF/F values of all frames from each recording are then concatenated into a matrix, e.g. `stackableFrameFluor_u50(1,2)` would contain the ΔF/F value for the second frame of the first recording within the U50 treatment group (lines 93-105). The average across groups for each frame is calculated using the `mean` function and the standard deviation is calculated using the `std` function (lines 129-130). The standard error is calculated by dividing the standard deviation by the square root of the number of recordings in the given group (line 131).

COMMUNICATION

AVERAGE ACROSS GROUPS PLOT

To plot the average fluorescence across a treatment group for the full time series, the mean calculated in line 133 is plotted as a solid line using `plot` (line 138). The standard error is displayed using the `fill` / `flip` functions and filled with a transparent color using the `alpha` function (lines 139-140). This is repeated for the control treatment group and displayed on the same graph (lines 142-144).

CELL TRACKING PLOT

To create a plot that displays the fluorescence of each cell in a given recording across frames, the average cell fluorescence calculated in the `quantifyCells` function is sent to the `cellFluor2Plot` function (Cell Quantifier, line 62). Within `cellFluor2Plot` a matrix containing the cell fluorescent values of each identified cell in the recording is created (Quantification Functions, lines 410-417). A plot is then created in which the fluorescence of a given cell across all frames is plotted as a single color; each cell's fluorescence is plotted as a different color on the same plot (Cell Quantifier, lines 67-79). The image generated is then formatted and saved to the working directory (lines 81-86).

RESULTS

CELL REGION IDENTIFICATION

The computational methods described above allowed for automated and unbiased handling, processing, analysis, and communication of the live cell imaging data. To automate the cell identification, tracking, and quantification process, a program was developed to identify cell regions (Figure 19). Raw images showing HyPerRed fluorescence in mycKOR-expressing HEK293 cells from the recording session (Figure 19.a) were sent into the program where cell regions were identified from the background (Figure 19.b). The identified region borders were then overlaid onto original image to quantify the fluorescence within the identified regions (Figure 19.c).

By automating the cell identification process, hundreds of images are able to be processed and quantified automatically within minutes, as opposed to the current method of outlining cells by hand to be quantified by the computer, which can take hours to days. In addition, outlining cells by hand allows room for human bias and error. While still a preliminary tool, the automated process allows for unbiased and rapid identification of cell regions for quantification. To improve the cell identification process, a program that can properly identify single cells within clusters would allow for higher resolution of analysis.

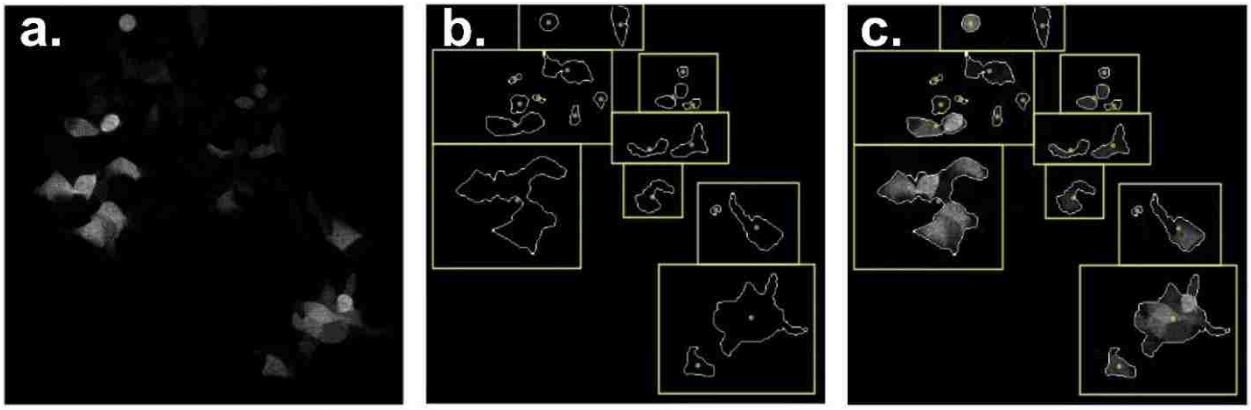

Figure 19: Cell Region Identification

(a) Sample image showing HyPerRed fluorescence in cells during real-time microscopy. **(b)** Cell borders identified by the cell identification algorithm with dots showing centroid calculation (no watershedding). **(c)** Overlay of border mask on original image.

To track the regions across the recording session, the centroids of each region were calculated (Figure 19.b-c, yellow dots). The centroids across all frames were clustered to statistically determine which centroids were likely to belong to the same cells. Each cell could then be tracked and quantified for its change in fluorescence from baseline and plotted over the duration of the recording (Figure 20).

For all recording days, four positions were recorded with two positions receiving a control vehicle serum and two positions receiving the experimental drug, e.g. U50,488. Figure 20 shows the change in fluorescence from a baseline from a sample session of cell regions receiving vehicle (Figure 20.a) and for cell regions receiving bath application of U50,488 (Figure 20.b). At the end of every recording, hydrogen peroxide was applied in the bath as a response control causing the spike in fluorescence observed at the end of the session. Across all sessions of cells receiving vehicle, 9% of cell regions had an increase in fluorescence over a threshold of 25%, and 11% of cell regions had a decrease in fluorescence under a threshold of -25% (Figure 20.c). Across all sessions of cells receiving 100nM of U50,488, 23% of cell regions had an increase in fluorescence over a threshold of 25%, and only 5% of cell regions had a decrease in fluorescence under a threshold of -25% (Figure 20.d). These results suggest that U50,488 tends to cause an increase in ROS production when compared to controls.

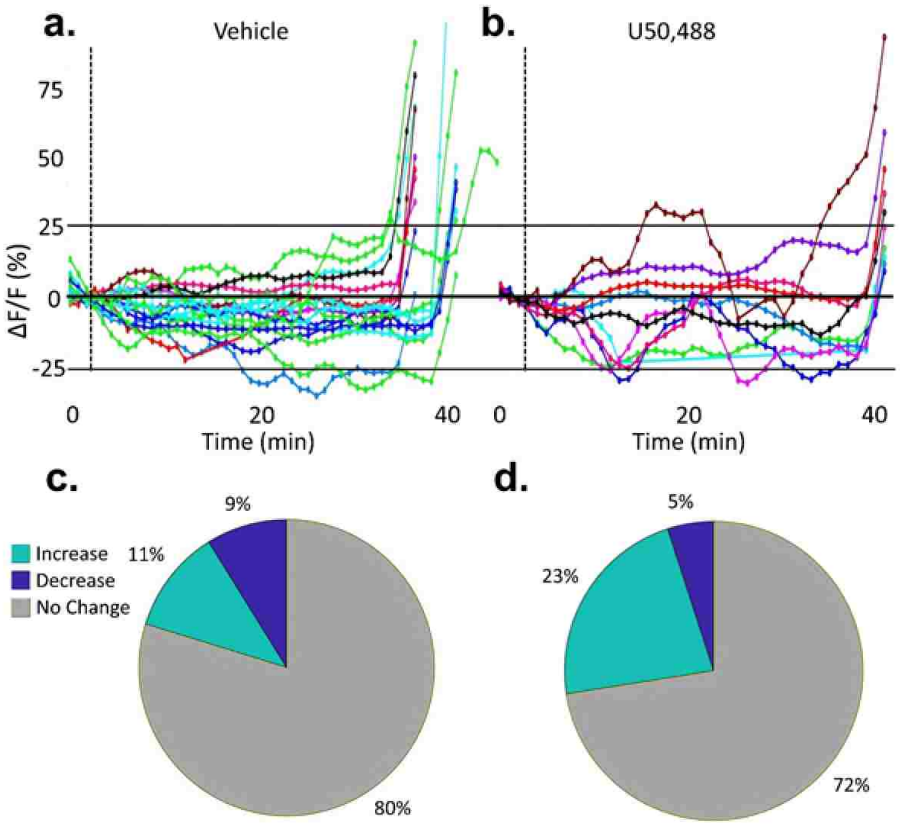

Figure 20: Cell Region Tracking and HyPerRed Fluorescence Quantification

Sample cell tracking plots of cells treated with vehicle **(a)** or U50,488 **(b)**. Each line represents the change in fluorescence of one identified cell within the frame. **(c)** Across all vehicle treated cells, 11% had a maximum increase in fluorescence (prior to hydrogen peroxide treatment) over a 25% threshold and 9% decreased below -25%. **(d)** Across all U50.488 treated cells, 23% had a maximum increase in fluorescence (prior to hydrogen peroxide treatment) over a 25% threshold and 5% decreased below -25%.

To investigate this observed phenomenon and to account for variability across cells, the average change in fluorescence across all cells was calculated using full-frame fluorescence. When the result of the full-frame analysis was averaged across the recording time period for all treatment groups, it showed that 100 nM U50,488 elevated ROS at approximately 30 minutes post-treatment and remained elevated at 45 min post-treatment (red) compared to controls (black) (Figure 21.a). This supports the evidence that KOR agonists increase pJNK to produce ROS. The time course of ROS activation also suggests that ROS generation is mediated through the earlier, arrestin-independent phase of JNK activation.

Additionally, the long-lasting elevation in ROS was blocked by pretreatment with 10 µM naloxone (black), and treatment with 10 µM naloxone 30 minutes after U50,488 activation completely reversed the increase in ROS within 15 min after the addition of naloxone (red) (Figure 21.b). This suggests that ROS activation is both a blockable and reversible process, which supports the hypothesis that ROS activation can be suppressed by arrestin-dependent activation of p38 MAPK by blocking JNK-activated signaling (Schattauer et el., 2019, ahead of print).

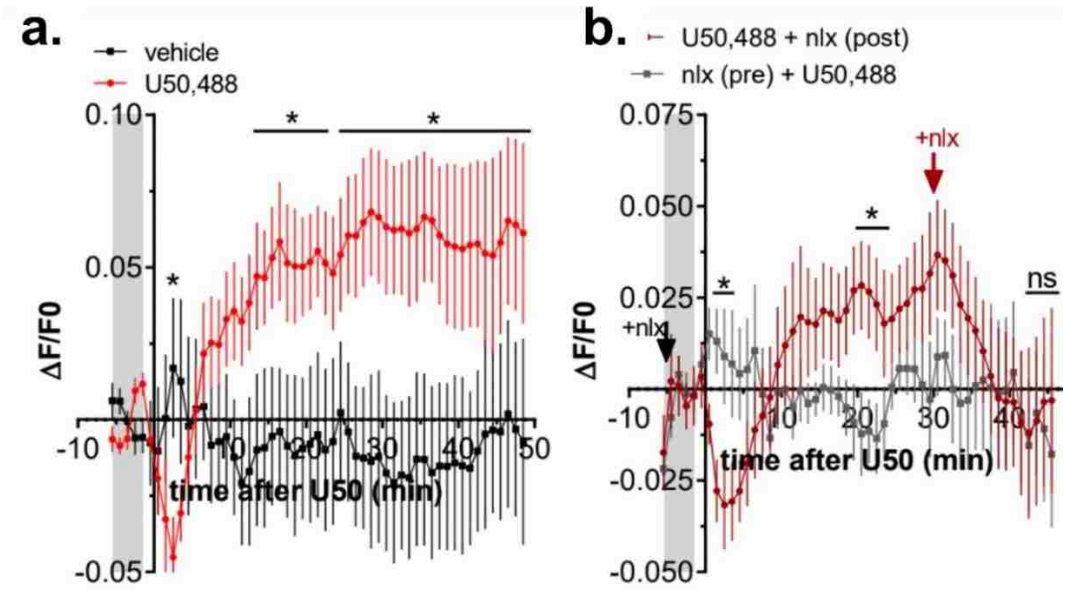

Figure 21: Effect of U50 and Naloxone on ROS Production *In Vitro*

(a) Change in fluorescence of cells treated with 100 nM U50,488 (red) or vehicle (black). Fluorescence was significantly increased by U50,488 in mycKOR-expressing HEK293 cells. Repeated measures two-way ANOVA (significant effect of time, P<0.0001 and significant interaction between time and drug, P<0.0001, n=9), with Holm-Sidak post-hoc comparison against vehicle (*P<0.05). Graph depicts mean+SEM. **(B)** Change in fluorescence of cells pretreated with 10 µM naloxone (black) or vehicle (red) 5 min prior to treatment with 100 nM U50,488; cells were post-treated with vehicle (black) or 10 µM naloxone (red) 30 min after U50,488. 45 min after U50,488, cells were treated with 200 µM H_2O_2. Fluorescence was significantly increased after U50,488 in mycKOR expressing HEK293 cells which were not pretreated with naloxone. Repeated measures two-way ANOVA (significant interaction between time and drug, P<0.0001, n=7), with Holm-Sidak post-hoc comparison against naloxone pretreatment (*P<0.05). Adapted from Schattauer et al., 2019, ahead of print.

A manuscript describing my findings has been submitted for review: Selena S. Schattauer, Andrea Bedini, Floyd Summers, Aiden Reilly-Treat, Mackenzie M. Andrews, Zeena M. G. Rivera, Jennifer S. Steger, Benjamin B. Land, Charles Chavkin (2019) Ligand directed activation of c Jun Kinase signaling by Gi/o protein coupled receptor agonists revealed by fluorescent sensors.

CONCLUSION

The principal findings of the studies presented in this chapter provide insight into the effects of KOR activation on cellular pathways and temporal sensitivity. The real-time cellular microscopy recordings demonstrate that pharmacological KOR activation results in arrestin-independent production of ROS which can be reversed by KOR antagonism and inhibited by arrestin-dependent p38 activity. This ROS production results in permanent KOR inactivation which may explain a mechanism of tolerance to repeated opioid receptor activation. Additionally, this KOR-dependent ROS production may contribute to the stress-induced vulnerability to neurodegeneration, mood disorders, and cardiovascular disease.

CHAPTER 5 – SIGNIFICANCE AND FUTURE DIRECTIONS

OVERVIEW

Chapter 5 summarizes the experimental findings described in the previous chapters within the context of the relationships between stress, cognition, and behavior. The significance of the computational work contributing to these findings is discussed with respect to both the broader implications on biological research as well as the implications on the experiments discussed herein. Within this context, a detailed description of the new findings is provided including their significance for future research and clinical considerations.

From the new findings, confounds with previous work and findings requiring further investigation are described. Future directions and descriptions of continuing experiments are provided including experimental suggestions for behavioral probing, pilot studies for extending cellular finding to animal models, and hardware design for probing of circuit dynamics.

Finally, concluding remarks on the greater relevance of these findings are provided and a call-to-action for the standardization of computational approaches is emphasized.

SUMMARY OF FINDINGS

The principal findings of these studies describe critical relationships between KOR activation and clinically relevant physiological changes at the behavioral, circuit, and cellular levels. In Chapter 2, two behavioral paradigms proved valuable in probing the effects of KOR activation on motivated behavior. The results demonstrated that KOR activation causes cognitive disruptions that affect both inhibitory control as well as working memory-dependent decision making. These findings provide insights into clinically relevant relationships between stress and cognitive and behavioral disorders. In Chapter 3, *in vivo* neural recordings illuminated a key finding about the effect of KOR activation on dopamine neuron activity. These results demonstrated that stress causes a KOR dependent increase in dopamine neuron activity in the VTA, a major reward center, following the stress exposure. These findings provide insights into a mechanism that may explain stress-induced changes in reward processing including the reward potentiating effects of stress on drugs of abuse. Finally, in Chapter 4, real-time cellular measurements elucidated a new branch of KOR activated signaling. These results demonstrated that KOR activation results in reversible production of ROS which leads to permanent KOR inactivation. These findings may explain both mechanisms of tolerance to opioid exposure as well as stress-induced vulnerability to neurodegeneration, mood disorders, and cardiovascular disease.

These findings provide insight into the role of the KOR-dynorphin system on the relationship between stress, cognition, and behavior. Investigating these relationships from the behavioral, circuit, and cellular levels, develops a framework of clinically relevant behavioral effects backed by the neural activity that may be contributing to those behavioral outputs, and provides cellular mechanisms underlying those neuronal changes that may act as pharmacological targets for the development of future treatments.

COMPUTATIONAL SIGNIFICANCE

The results and analyses presented in these studies would not have been possible without the development of novel computational approaches and tools. As discussed in Chapter 1, advances in the state-of-the-art investigational tools are allowing researchers to probe more complex questions at higher informational resolutions. Resultantly, these tools are producing increasingly larger and more complex data sets. However, these technologies have been advancing so rapidly, that the classical analysis methods are struggling to capture the breadth of information available in the new data sets.

As a result, there is a major bottleneck in biological research between data acquisition and data analysis, primarily due to the heterogeneity and multivariability of biological systems. Further complicating the problem, biological research is relatively decentralized – there are thousands of biology labs developing new experiments and probing systems in new ways. Thus, standardizing the analytic methods for this vastly heterogenous discipline is a daunting yet critical task.

In order to 1) capture the increased complexity and resolution of data, 2) account for heterogeneity and multivariability, and 3) standardize analysis methods across the biological discipline, new tools must make considerations at four major steps in the analysis process: data handling, data processing, data analysis, and data communication. In the preceding chapters, the considerations and methods used at each of these four steps were detailed and the code used to develop the tools described was documented.

Specific effort was devoted to incorporating the classical analysis techniques for each type of experiment, while building off those methods to fill the analysis gap needed to answer the posed research questions. In the case of the behavioral experiments, classical analyses reduce the dimensionality of the complex temporal information. In order to dissociate the factors affecting behavior, effort was devoted to developing tools to analyze and visualize the micro and macro temporal components of motivated behavior. Such tools included database structuring, burst analysis, latency analyses, training observations, and raster plot generation.

In the case of the circuit level interrogations, classical analyses down-sample data and only explore discrete changes in signal qualities. In order to unravel the neural activity and outputs contributing to disruptions in circuit dynamics during behavior, effort was devoted to developing tools to preserve the temporal resolution of the data while expanding the types of analyses explored. Such tools included signal filtering and normalization, Fourier analysis, and temporally comprehensive signal comparisons.

Finally, in the case of cellular imaging, classical analyses require extensive involvement of the researcher which is time consuming and leaves room for biasing and arbitration of results. In order to efficiently and appropriately analyze real-time cellular microscopy images, effort was devoted to developing tools to automate the cellular identification, tracking, and quantification steps. Such tools included image analysis for cell identification, centroid calculation and statistical clustering for cell tracking, and automated cellular quantification from boundary masks.

NEW UNDERSTANDING

Due to the intellectual curiosity of the researchers, advancements of investigational technologies, and the development of the computational tools described previously, the studies detailed have started to fill the knowledge gaps introduced in Chapter 1 (Figure 2 v. Figure 22). At the behavioral level, the DRL task confirmed the hypothesized link between stress-mediated KOR-activation and disruptions of inhibitory control (Figure 22.a). These studies found that both stress-induced and pharmacologically-induced KOR activation cause reductions in DRL performance and increases in burst responses that are both VTA-dependent and arrestin-independent. The operant delayed alternation task challenged the existing information in the field by demonstrating that pharmacologically-induced KOR activation disrupts the performance of a working memory dependent task in a PFC-dependent manor (Figure 22.b. However, repeated forced swim stress was not sufficient to produce behavioral disruptions through the PFC circuit being interrogated. Additionally, spontaneous morphine withdrawal actually increased performance on the task, a finding that requires further investigation. One hypothesis is that the enhanced activity is due to compensatory inputs from other brain regions.

At the circuit level, *in vivo* GCaMP fiber photometry revealed that pharmacological KOR activation does cause short-term decreased in dopamine neuron activity in the VTA, consistent with previous findings (Figure 22.c). However, both pharmacological KOR activation and stress exposure appeared to cause post-inhibitory or post-stress increases in VTA DAT activity. The post-stress increases in activity were compounded following repeated stress exposures. This finding may begin to provide a neural framework for reward potentiation, i.e. KOR-activation-induced suppression of VTA DAT activity may lead to a post-suppression increase in neural activity consistent with a reward or relief signal. Further investigation may be needed to determine whether this finding supports or contradicts the hedonic-state hypothesis.

Finally, at the cellular level, real-time cellular microscopy helped to elucidate the mechanisms and physiological effects of KOR agonist-induced increases in JNK production (Figure 22.d). These findings demonstrated that KOR agonism causes short-term arrestin-independent increases in JNK production which leads to increases in ROS production and KOR inactivation. However, KOR agonism also results in a slower, arrestin-dependent activation of p38 which suppresses JNK production, ultimately reducing ROS production. This supports the previous finding that KOR antagonists, such as nor-BNI, are actually JNK-biased agonists and the long-lasting nature of their action is due to the ROS-induced permanent inactivation of KOR. Parsing out this third

branch of GPCR signaling, and understanding how these effects play out *in vivo* may allow for the development of more targeted therapies for pain and mood disorders.

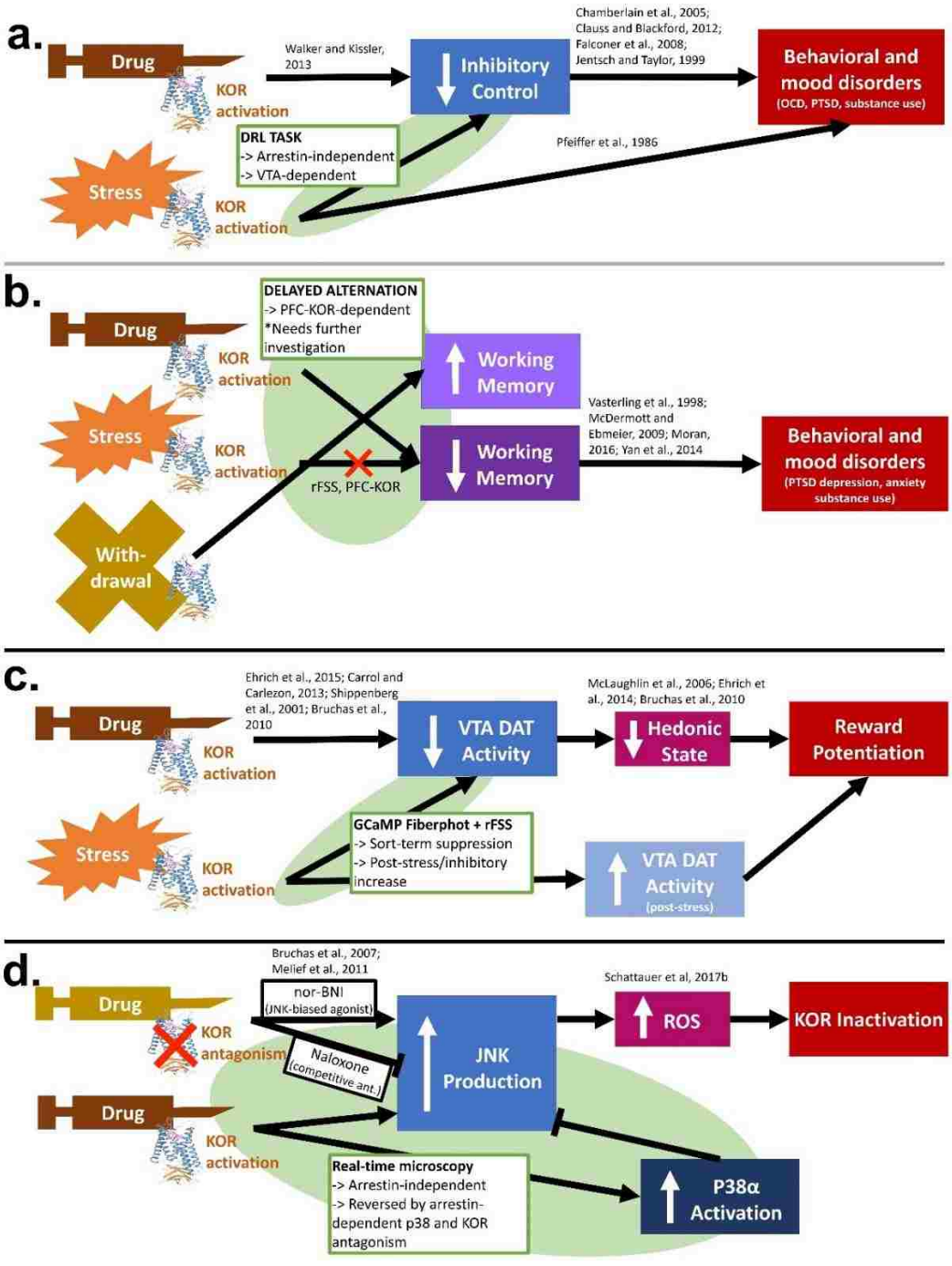

Figure 22: State of Understanding Incorporating New Findings

Schematics depicting the new understanding of behavioral **(a-b)**, circuit **(c)**, and cellular **(d)** level relationships between KOR activation and respective outputs. Content within green ovals represents new knowledge gained from the experiments described. Arrow heads indicate an activating or causal relationship, blunted heads indicate a blocking or suppressatory relationship.

Note: Primarily intended as a reference for the information presented in the Chapters 2-4, compare to Figure 2.

FUTURE DIRECTIONS

WITHDRAWAL, STRESS, AND DELAYED ALTERNATION

The findings presented were helpful in filling some of the knowledge gaps in the field, however, some of the findings seemed to challenge existing knowledge or open the door to a new series of questions. As eluded to previously, the delayed alternation findings challenged both the existing work showing that stress-induced KOR activation disrupts working memory and the hypothesis that opioid withdrawal would disrupt working memory. One potential explanation for the lack of stress-effect is that swim stress does not cause dynorphin release in the PFC or that compensatory mechanisms from other brain regions reduce the observable behavioral effect. In order to detangle these hypotheses, future experiments may include: 1) testing other stress paradigms such as social defeat stress or fear-induced stress such as foot shock, 2) measuring stress-induced dynorphin release in the PFC, and 3) recording coordinated behavior between brain regions that project to the PFC to determine if compensatory effects could be at play.

Addressing the morphine withdrawal findings, potential explanations for the increase in performance are 1) the animals experienced an increase in motivation due to morphine-induced bowel disruptions leading to the animals being more food motivated, 2) the animals were less distracted due to a decrease in locomotion and thus were more focused on the task, or 3) compensatory effects from other regions during the morphine exposure resulted in increased performance during spontaneous withdrawal. To parse out what effects may be contributing to these observations, future experiments may include: 1) test performance following precipitated withdrawal to determine if compensatory effects are at play, 2) quantify animal locomotion during the task, and 3) test the withdrawal effects of other opioids and/or alternative administration protocols on the behavior paradigm.

HYPERRED FIBER PHOTOMETRY

While behavioral findings often require follow up experiments to probe lower level – circuit and cellular – effects to understand behavior confounds, cellular level findings require higher level experiments to investigate their applicability to system-level physiology. In the case of the real-time cellular microscopy findings, the time course and magnitude of the KOR-activation-induced ROS production, as well as the physiological consequences thereof, must be confirmed *in vivo*. In order to do so, the same fiber photometry techniques used to probe calcium signaling *in vivo* using GCaMP can be used to probe ROS production using HyPerRed.

To record ROS production following KOR-activation *in vivo*, pilot studies using HyPerRed fiber photometry have been initiated (Figure 23). Increasing doses of U50,488 were administered while the HyPerRed signal was recorded. While preliminary, this pilot appears to support the hypothesis that KOR activation increases ROS production *in vivo.*

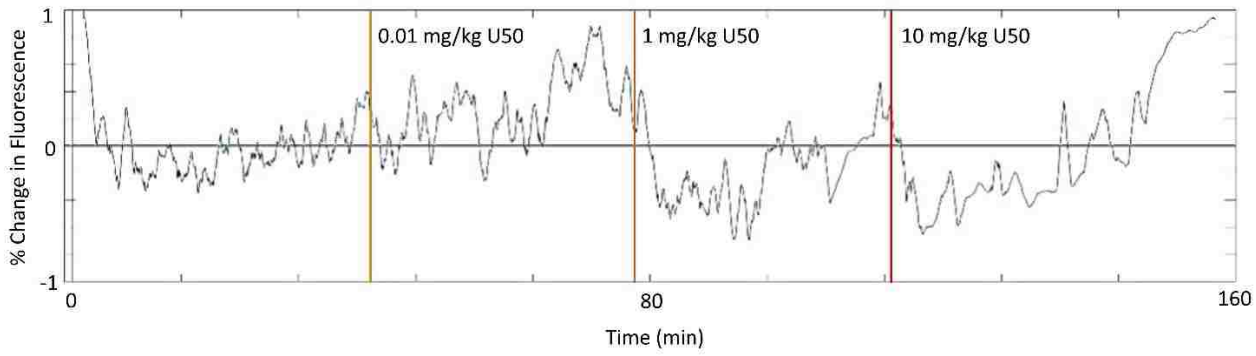

Figure 23: *In Vivo* HyPerRed Fiber Photometry Pilot

Sample trace of *in vivo* HyPerRed fiber photometry recordings during escalating doses of U50.488 administration. Doses of 0.01mg/kg (yellow), 1mg/kg (orange), and 10mg/kg (red) were administered approximately 40 minutes apart. Pilot data appears to show a trend toward an increase in HyPerRed fluorescence following U50,488 administration.

CONNECTING THE REGIONS – OPTRODE

While fiber photometry begins to probe circuit level questions, it is currently limited to recording from one brain region at a time. To investigate circuit dynamics, such as the coordination dynamics discussed in the Withdrawal, Stress, And Delayed Alternation section, the ability to record from multiple brain regions simultaneously it needed.

Multi-site recordings would help connect regional information such as the VTA-dependent cognitive disruptions investigated with the DRL task with the PFC-dependent disruptions investigated with the delayed alternation task, while providing a new dimension of temporal resolution to parse out coordination effects. However, multi-site recordings require technological advancements from both the hardware and software sides of the equation. From a hardware standpoint, multi-site recording devices are limited by the size and weight constraints of the animal – the animal must be able to support the device on their head during behavior. The design and development of such a device is detailed in Appendix I, Capstone Thesis. This work may provide the foundation for future advancements in neuronal recordings and expand the types of questions that can be pursued by researchers.

CONCLUDING REMARKS

This work marks an important step forward in our understanding of the relationship between stress, cognition, and behavior within the context of the KOR-dynorphin system. The findings presented improve our understanding of the clinically relevant physiological changes caused by stress and how stress contributes to cognitive, behavioral, and mood disorders such as pain, PTSD, anxiety, depression, and drug abuse. Additionally, this work provides insight into potential therapeutic targets to treat such disorders.

These findings would not have been possible without the computational approaches described herein. In order for neuroscience and larger biological disciplines to continue advancing, there is an ever-growing need to standardize and innovate computational approaches to analyzing heterogenous, multivariate systems. Without a consolidation of computational knowledge and standardization of approaches, the analysis bottleneck will continue to constrict, vastly limiting the rate at which we can understand and treat not just physical, but societal ailments.

ACKNOWLEDGEMENTS

KEY PERSONEL

I would like to thank Dr. Charles Chavkin for allowing me to work in his lab, for opening this field of research, and for inspiring the groundwork and direction for these projects. I would also like to extend a very special thank you to my direct mentor, Dr. Antony Abraham, for his consistent devotion to ensuring the academic and personal success of his students and for his generosity with time and knowledge throughout this project, you will have made an invaluable impact on myself and lives of every student you have mentored. Additionally, I want to thank all of the members of the Chavkin Lab for their consistent support, collaboration, and friendship. I also want to acknowledge key members of the Zweifel Lab: Larry Zweifel, Yong Jo, and Barbara Juarez, for their collaboration and sharing of knowledge, workspace, and materials. Finally, I want to thank my bioengineering committee chair, Dr. Andre Berndt, for his willingness to step in as chair of my thesis committee.

EQUIPMENT, FACILITIES, AND SOFTWARE

The majority of this project was conducted in the University of Washington Health Sciences Building. All of the experimental design and methods were performed in Chavkin Lab research spaces. The computational work was conducted on my personal machine (Dell XPS 15 9550, Microsoft Windows 10 Education) with an personal education license of Mathworks MATLAB software.

FUNDING

I would like to thank the National Institute of Mental Health, the National Institute on Drug Abuse, the UW Institute for Neural Engineering, the Washington Research Foundation, and the Levinson Emerging Scholar Award for funding the materials and spaces for this work. I would also like to thank the UW Computational Neuroscience Program for funding my research time and efforts. (R01 DA030074 (CC), T32 DA007278 (AA), and P50MH106428 (LZ, CC). The authors declare no financial conflict of interest.)

REFERENCES

Abbott LF, Regehr WG. Synaptic computation. Nature. 431(7010):796-803.

Abraham AD, Fontaine HM, Song AJ, Andrews MM, Baird MA, Brigitte LK, Land BB, Chavkin C. (2018) κ-Opioid Receptor Activation in Dopamine Neurons Disrupts Behavioral Inhibition. Neuropsychopharmacology. 43:362-372.

Alyass A, Turcotte M, Meyre D. (2015) From big data analysis to personalized medicine for all: challenges and opportunities. BMC Medical Genomics. 8:33.

Arce E, Santisteban C. (2006) Impulsivity: a review. Psicothema. 18(2):213-220.

Atkinson JW, Birch D. (1978) An introduction to motivation (2nd ed.). New York: Van Nostrand.

Bliss TV, Collingridge GL. (1993) A synaptic model of memory: long-term potentiation in the hippocampus. Nature. 361(6407):31-39.

Bockaert J, Pin JP. (1999) Molecular tinkering of G protein-coupled receptors: an evolutionary success. EMBO J. 18(7):1723-9.

Brown SW, Johnson TM, Sohl ME, Dumas MK. (2015) Executive attentional resource in timing: Effects of inhibitory control and cognitive aging. Journal of Experimental Psychology: Human Perception and Performance. 41(4):1063-1083.

Bruchas MR, Chavkin C. (2010) Kinase cascades and ligand-directed signaling at the kappa opioid receptor. Psychopharmacology (Berl). 210:137-147.

Bruchas MR, Land BB, Chavkin C. (2010) The dynorphin/kappa opioid system as a modulator of stress-induced and pro-addictive behaviors. Brain Res. 1314:44-55.

Bruchas MR, Macey TA, Lowe JD, Chavkin C. (2006) Kappa opioid receptor activation of p38 MAPK is GRK3- and arrestin-dependent in neurons and astrocytes. J. Biol. Chem. 281:18081-18089.

Bruchas MR, Yang T, Schreiber S, Defino M, Kwan SC, Li S, Chavkin C. (2007) Long-acting kappa opioid antagonists disrupt receptor signaling and produce noncompetitive effects by activating c-Jun N-terminal kinase. J. Biol. Chem. 282(41):29803-29811.

Brust TF, Morgenweck J, Kim SA, Rose JH, Locke JL, Schmid CL, Zhou L, Stahl EL, Cameron MD, Scarry SM, Aubé J, Jones SR, Martin TJ, Bohn LM. (2016) Biased agonists of the

kappa opioid receptor suppress pain and itch without causing sedation or dysphoria. Science Signaling. 9(456):ra117.

Campeau S, Liberzon I, Morilak D, Ressler K. (2011) Stress modulation of cognitive and affective processes. Stress. 14:503–19.

Campioni MR, Xu M, McGehee DS. (2009) Stress-induced changes in nucleus accumbens glutamate synaptic plasticity. J. Neurophysiol. 101:3192–3198.

Carey AN, Lyons AM, Shay CF, Dunton O, McLaughlin JP. (2009) Endogenous kappa opioid activation mediates stress-induced deficits in learning and memory. J Neurosci. 29(13):4293-300.

Carlezon WA Jr, Béguin C, DiNieri JA, Baumann MH, Richards MR, Todtenkopf MS, Rothman RB, Ma Z, Lee DY, Cohen BM. (2006) Depressive-like effects of the kappa-opioid receptor agonist salvinorin A on behavior and neurochemistry in rats. J. Pharmacol. Exp. Ther. 316(1):440-447.

Carroll FI, Carlezon WA Jr. (2013) Development of κ opioid receptor antagonists. J Med. Chem. 56(6):2178-95.

Chamberlain SR, Blackwell AD, Fineberg NA, Robbins TW, Sahakian BJ. (2005) The neuropsychology of obsessive compulsive disorder: the importance of failures in cognitive and behavioural inhibition as candidate endophenotypic markers. Neurosci Biobehav Rev. 29:399–419.

Chavkin C, Martinez D. (2015) Kappa Antagonist JDTic in Phase 1 Clinical Trial. Neuropsychopharmacology. 40(9):2057-2058.

Clauss JA, Blackford JU. (2012) Behavioral Inhibition and Risk for Developing Social Anxiety Disorder: A Meta-Analytic Study. Journal of the American Academy of Child & Adolescent Psychiatry. 51(10):1066-1075.

Cowan N. (2009) What are the differences between long-term, short-term, and working memory? Prog. Brain Res. 169:323-338.

Cui G, Jun SB, Jin X, Pham MD, Vogel SS, Lovinger DM, Costa RM. (2013) Concurrent activation of striatal direct and indirect pathways during action initiation. Nature. 494:238–242.

Destexhe A, Marder E. (2004) Pasticity in single neuron and circuit computations. Nature. 431(7010):789-795.

Dudchenko PA. (2004) An overview of the tasks used to test working memory in rodents. Neuroscience and Behavioral Reviews. 28:699-709.

Ehrich JM, Messinger DI, Knakal CR, Kuhar JR, Schattauer SS, Gruchas MR, Zweifel LS, Kieffer BL, Phillips PEM, Chavkin C. (2015) Kappa Opioid Recpetor-Induced Aversion Requires p38 MAPK Activation in VTA Dopamine Neurons. J Neurosci. 35(37):12917-12931.

Ehrich JM, Phillips PEM, Chavkin C. (2014) Kappa Opioid Receptor Activation Potentiates the Cocaine-Induced Increase in Evoked Dopamine Release Recorded *In Vivo* in the Mouse Nucleus Accumbens. Neuropsychopharmacology. 39(13):3036-3048.

Ermakova YG, Bilan DS, Matlashov ME, Mishina NM, Markvicheva KN, Subach OM, Subach FV, Bogeski I, Hoth M, Enikolopov G, Belousov VV. (2014) Red fluorescent genetically encoded indicator for intracellular hydrogen peroxide. Nature Communications. 5:5222.

Falconer E, Bryant R, Felmingham KL, Kemp AH, Gordon E, Peduto A, Olivieri G, Williams LM. (2008) The neural networks of inhibitory control in posttraumatic stress disorder. J Psychiatry Neurosci. 33(5):413-422.

Flor H1, Nees F. (20140) Learning, memory and brain plasticity in posttraumatic stress disorder: context matters. Restor. Neurol. Neurosci. 32(1):95-102.

Fornaro M, Gabrielli F, Albano C, Fornaro S, Rizzato S, Mattei C, Solano P, Vinciguerra V, Fornaro P. (2009) Obsessive-compulsive disorder and related disorders: a comprehensive survey. Ann Gen Psychiatry. 8:13.

Fox HC, Bergquist KL, Hong KI, Sinha R. (2007) Stress-induced and alcohol cue-induced craving in recently abstinent alcohol-dependent individuals. Alcohol Clin. Exp. Res. 31:395–403.

Fraga MF, Ballestar E, Paz MF, Ropero S, Setien F, Ballestar ML, Heine-Suner D, Cigudosa JC, Urioste M, Benitez J, Boix-Chornet M, Sanches-Aguilera A, Ling C, Carlsson E, Poulsen P, Vaag A, Stephan Z, Spector TD, Wu Y-Z, Plass C, Esteller M. (2005) Epigenetic differences arise during the lifetime of monozygotic twins. Proceedings of the National Academy of Sciences. 102:10604-10609.

Gold PW, Chrousos GP. (2002) Organization of the stress system and its dysregulation in melancholic and atypical depression: high vs low CRH/NE states. Mol Psychiatry. 7:254–275.

Gold PW, Goodwin FK, Chrousos GP. (1988) Clinical and biochemical manifestations of depression: relation to the neurobiology of stress (Part 2 of 2 parts). N Engl J Med. 319:348–353.

Granier S, Kobilka B. (2012) A new era of GPCR structural and chemical biology. Nat. Chem. Biol. 8(8):670-673.

Gunaydin LA, Grosenick L, Finkelstein JC, Kauvar IV, Fenno LE, Adhikari A, Lammel S, Mirzabekov JJ, Airan RD, Zalocusky KA, Tye KM, Anikeeva P, Malenka RC, Deisseroth K. (2014). Natural neural projection dynamics underlying social behavior. Cell. 157:1535–1551.

Hart AS, Rutledge RB, Glimcher PW, Phillips PEM. (2014) Phasic dopamine release in the rat nucleus accumbens symmetrically encodes a reward prediction error term. J Neurosci. 34:698–704.

Hayashi-Takagi A, Takaki M, Graziane N, Seshadri S, Murdoch H, Dunlop AJ, Makino Y, Seshadri AJ, Ishizuka K, Srivastava DP, Xie Z, Baraban JM, Houslay MD, Tomoda T, Brandon NJ, Kamiya A, Yan Z, Penzes P, Sawa A. (2010) Disrupted-in-Schizophrenia 1 (DISC1) regulates spines of the glutamate synapse via Rac1. Nat Neurosci. 13(3):327-32.

Hebb DO. (1949) Organization of Behavior. New York: Wiley.

Holtmaat A, Svoboda K. (2009) Experience-dependent structural synaptic plasticity in the mammalian brain. Nat Rev Neurosci. 10(9):647-58.

Horwood JM, Ripley TL, Stephens DN. (2001) DRL performance in mice with deletion of tPA, uPA or PAI-1 genes. Behav Pharmacol. 12:487–496.

Izaki Y, Maruki K, Hori K, Nomura M. (2001) Effect of rat medial prefrontal cortex temporal inactivation on a delayed alternation task. Neuroscience Letters. 315(3):129-132.

Jentsch JD, Taylor JR. (1999) Impulsivity resulting from frontostriatal dysfunction in drug abuse: implications for the control of behavior by reward-related stimuli. Psychopharmacology (Berl). 146:373–390.

Kam AY, Chan AS, Wong YH. (2004) Kappa-opioid receptor signals through Src and focal adhesion kinase to stimulate c-Jun N-terminal kinases in transfected COS-7 cells and human monocytic THP-1 cells. J. Pharmacol. Exp. Ther. 310:301-310 (2004).

Kirkpatrick K, Marchall AT, Clarke J, Cain ME. (2013) Environmental Rearing Effects on Impulsivity and Reward Sesitivity. Behav. Neurosci. 127(5):712-724.

Knoll AT, Carlezon WA Jr. (2010) Dynorphin, stress, and depression. Brain Res. 1314:56-73.

Koob GF, Le Moal M. (1997) Drug abuse: hedonic homeostatic dysregulation. Science. 278:52.

Kreek MJ, Koob GF. (1998) Drug dependence: stress and dysregulation of brain reward pathways. Drug Alcohol Depend. 51:23–47.

Kuzmin A, Madjid N, Terenius L, Ogren SO, Bakalkin G. (2006) Big dynorphin, a prodynorphin-derived peptide produces NMDA receptor-mediated effects on memory, anxiolytic-like and locomotor behavior in mice. Neuropsychopharmacology. 31(9):1928-1937.

Land BB, Bruchas MR, Lemos JC, Xu M, Melief EJ, Chavkin C. (2008) The dysphoric component of stress is encoded by activation of the dynorphin kappa-opioid system. J Neurosci. 28:407–414.

Lazarus, RS. (1982) Thoughts on the relations between emotion and cognition. American Psychologist, 37, 1019–1024.

Lazarus RS. (1999) Stress and Emotion: A New Synthesis. Springer Publishing Company; New York.

LeDoux JE. (2000). Emotion circuits in the brain. Annual Review of Neuroscience, 23, 155-184.

Leventhal H, Cleary PD. (1980) The smoking problem: a review of the research and theory in behavioral risk modification. Psychol Bull. 88:370–405.

Li J-G, Luo L-Y, Krupnick JG, Benovic JL, Liu-Chen L-Y. (1999) U50,488H-induced Internalization of the Human k Opioid Receptor Involves a b-Arrestin- and Dynamin-dependent Mechanism. Journal of Biological Chemistry. 274(17):12087-12094.

Lieberman, DA, (2004) Learning and memory: An integrative approach, Belmont, CA.: Wadsworth/Thompson Learning.

Liu W, Ge T, Leng Y, Pan Z, Fan J, Yang W, Cui R. (2017) The Role of Neural Plasticity in Depression: From Hippocampus to Prefrontal Cortex. Neural Plasticity. 2017: 6871089.

Lorenz K. (1950) The comparative method of studying innate behavior patterns. In Symposia of the Society of Experimental Biology. Cambridge: Cambridge University Press.

Luethi M, Meier B, Sandi C. (2008) Stress Effects on Working Memory, Explicit Memory, and Implicit Memory for Neutral and Emotional Stimuli in Healthy Men. Front. Beh. Neurosci. 2:5.

Lutz PE, Ayranci G, Chu-Sin-Chung P, Matifas A, Koebel P, Filliol D, Befort K, Ouagazzal A-M, Kieffer BL. (2014) Distinct Mu, Delta, and Kappa Opioid Receptor Mechanisms Underlie Low Sociability and Depressive-Like Behaviors During Heroin Abstinence. Neuropsychopharmacology. 39(11):2694-2705.

Mague SD, Pliakas AM, Todtenkopf MS, Tomasiewicz HC, Zhang Y, Stevens WC Jr, Jones RM, Portoghese PS, Carlezon WA Jr. (2003) Antidepressant-like effects of kappa-opioid receptor antagonists in the forced swim test in rats. J. Pharmacol. Exp. Ther. 305(1):323-330.

Mao T, O'Connor DH, Scheuss V, Nakai J, Svoboda K. (2008) Characterization and Subcellular Targeting of GCaMP-Type Genetically-Encoded Calcium Indicators. PLoS ONE. 3(3):e1796.

Marder E, Thirumalai V. (2002) Cellular, synaptic and network effects of neuromodulation. Neural Netw. 15:479–493.

Margolis EB, Lock H, Chefer VI, Shippenberg TS, Hjelmstad GO, Fields HL. (2006) Kappa opioids selectively control dopaminergic neurons projecting to the prefrontal cortex. Proc. Nat'l. Acad. Sci. USA. 103(8):2938-42.

Marlatt GA, Gordon JR. (1985) Relapse prevention: maintenance strategies in the treatment of addictive behaviors. Guilford Press, New York.

Marx V. (2013) The big challenges of big data. Nature. 498:255-260.

Mattmann CA. (2013) A vision for data science. Nature. 493:473-475.

McDermott LM, Ebmeier KP. (2009) A meta-analysis of depression severity and cognitive function. J Affect Disord. 119:1–8.

McDougall W. (1970) The nature of instincts and their place in the constitution of the human mind. In W. A. Russell (Ed.), Milestones in motivation. New York: Appleton-Century-Crofts.

McEwen BS. (2000) The neurobiology of stress: from serendipity to clinical relevance. Brain Res. 886(1-2):172-189.

McGuire PS, Seiden LS. (1980) The effects of tricyclic antidepressants on performance under a differential-reinforcement-of-low-rates schedule in rats. J Pharmacol. Exp. Ther. 214(3):635-641.

McLaughlin JP, Li S, Valdez J, Chavkin TA, Chavkin C. (2006) Social defeat stress-induced behavioral responses are mediated by the endogenous kappa opioid system. Neuropsychopharmacology. 31(6):1241-1248.

McLaughlin JP, Marton-Popovici M, Chavkin C. (2003) Kappa opioid receptor antagonism and prodynorphin gene disruption block stress-induced behavioral responses. J Neurosci. 23:5674–5683.

McLennan GP, Kiss A, Miyatake M, Belcheva MM, Chambers KT, Pozek JJ, Mohabbat Y, Moyer RA, Bohn LM, Coscia CJ. (2008). Kappa opioids promote the proliferation of astrocytes via Gbetagamma and beta-arrestin 2-dependent MAPK-mediated pathways. J. Neurochem. 107(6):1753-1765.

Melief EJ, Miyatake M, Carroll FI, Béguin C, Carlezon WA Jr, Cohen BM, Grimwood S, Mitch CH, Rorick-Kehn L, Chavkin C. (2011) Duration of action of a broad range of selective κ-opioid receptor antagonists is positively correlated with c-Jun N-terminal kinase-1 activation. Nol. Pharmacol. 80(5):920-929.

Miller GA, Galanter E, Pribram KH. (1960) Plans and the structure of behavior. New York: Holt, Rinehart and Winston, Inc.

Mischel W, Ebbesen EB, Raskoff Zeiss A. (1972) Cognitive and attentional mechanisms in delay of gratification. Journal of Personality and Social Psychology, 21(2):204-218.

Moeller FG, Barratt ES, Dougherty DM, Schmitz JM, Swann AC. (2007) Psychiatric aspects of impulsivity. Am J Psychiatry. 158(11):1783-93.

Moore SA, Zoellner LA, Mollenholt N. (2008) Are expressive suppression and cognitive reappraisal associated with stress-related symptoms? Behav Res Ther. 46(9):993-1000.

Moran TP. (2016) Anxiety and working memory capacity: A meta-analysis and narrative review. Psychol. Bull. 142(8):831-864.

Morris SE, Cuthbert BN. (2012) Research Domain Criteria: cognitive systems, neural circuits, and dimensions of behavior. Dialogues Clin Neurosci. 14(1):29-37.

Moult PR, Corrêa SAL, Collingridge GL, Fitzjohn SM, Bashir ZI. (2008) Co-activation of p38 mitogen-activated protein kinase and protein tyrosine phosphatase underlies metabotropic glutamate receptor-dependent long-term depression. Journal of Physiology. 586(10):2499–2510.

Ninomiya Y, Zhao W, Saga Y. (2016) GBIQ: a non-arbitrary, non-biased method for quantification of fluorescent images. Nature Scientific Reports. 6:26454

O'Donnell JM, Marek GJ, Seiden LS. (2005) Antidepressant effects assessed using behavior maintained under a differential-reinforcement-of-low-rate (DRL) operant schedule. Neurosci Biobehav Rev. 29:785–798.

Ohno M, Yamamoto T, Ueki S. (1991) Effect of the kappa-receptor agonist, U-50,488H, on cerebral ischemia-induced impairment of working memory assessed in rats by a three-panel runway task. Eur. J. Pharmacol. 193(3):357-361.

Overmier JB, Seligman MEP. (1967) Effects of inescapable shock upon subsequent escape and avoidance learning. Journal of Comparative and Physiological Psychology, 63, 23–33.

Paris JJ, Railley KJ, McLaughlin JP. (2011) Kappa Opioid Receptor-Mediated Disruption of Novel Object Recognition: Relevance for Psychostimulant Treatment. J. Addict. Res. Ther. 4:007.

Pavlov, IP. (1960) Conditioned reflexes. New York: Dover.

Petri H. (2005) Four motivational components of behavior. REME. ISSN 1138-493X, Vol. 8, No. 20-21.

Pfeiffer A, Brantl V, Herz A, Emrich HM. (1986) Psychotomimesis mediated by kappa opiate receptors. Science. 233(4765):774-776.

Platkiewicz J, Brette R. (2010) A Threshold Equation for Action Potential Initiation. PLoS Comput. Biol. 6(7):e1000850.

Piazza PV, Deminiere JM, Le Moal M, Simon H. (1990) Stress- and pharmacologically induced behavioral sensitization increases vulnerability to acquisition of amphetamine self-administration. Brain Res. 514:22–26.

Pitcher JA, Freedman NJ, Lefkowitz RJ. (1998) G protein-coupled receptor kinases. Annual Rev. Biochem. 67:653-692.

Rossi MA, Hayrapetyan VY, Maimon B, Mak K, Je HS, Yin HH. (2012) Prefrontal cortical mechanisms underlying delayed alternation in mice. Journal of neurophysiology. 108:1211-22.

Russell JA, Mehrabian A. (1975) The mediating role of emotions in alcohol use. J Stud Alcohol. 36:1508–1536.

Russo SJ, Wilkinson MB, Mazei-Robison MS, Dietz DM, Maze I, Krishnan V, Renthal W, Graham A, Birnbaum SG, Green TA, Robison B, Lesselyong A, Perrotti LI, Bolaños

CA, Kumar A, Clark MS, Neumaier JF, Neve RL, Bhakar AL, Barker PA, Nestler EJ. (2009) Nuclear factor kappa B signaling regulates neuronal morphology and cocaine reward. J Neurosci.29(11):3529-37.

Schattauer SS, Bedini A, Summers F, Reilly-Treat A, Andrews MM, Rivera ZMG, Steger JS, Land BB, Chavkin C. (2019) Ligand directed activation of c Jun Kinase signaling by Gi/o protein coupled receptor agonists revealed by fluorescent sensors. Ahead of print.

Schattauer SS, Kuhar JR, Song AJ, Chavkin C. (2017a) Nalfurafine is a G-protein biased agonist having significantly greater bias at the human than rodent form of the kappa opioid receptor. Cell Signal. 32:59-65.

Schattauer SS, Land BB, Reichard KL, Abraham AD, Burgeno LM, Kuhar JR, Phillips PEM, Ong SE, Chanvkin C. (2017b) Peroxiredoxin 6 mediates Gαi protein-coupled receptor inactivation by cJun kinase. Nature Communications. 8:743

Schattauer SS, Miyatake M, Shankar H, Zietz C, Levin JR, Liu-Chen L, Gurevich VV, Rieder MJ, Chavkin C. (2012) Ligand Directed Signaling Differences between Rodent and Human κ-Opioid Receptors. Journal of Biological Chemistry. 287:41595-41607.

Schultz W, Dayan P, Montague PR. (1997) A neural substrate of prediction and reward. Science. 275:1593–1599.

Selleck RA, Lake C, Estrada V, Riederer J, Andrzejewski M, Sadeghian K, Baldo BA. (2015) Endogenous opioid signaling in the medial prefrontal cortex is required for the expression of hunger-induced impulsiveaction. Neuropsychopharmacology. 40:2464-2474.

Shaham Y, Stewart J. (1994). Exposure to mild stress enhances the reinforcing efficacy of intravenous heroin self-administration in rats. Psychopharmacology (Berl). 114:523–527.

Shiffman S. (1982) Relapse following smoking cessation: a situational analysis. J Consult Clin Psychol. 50:71–86.

Shippenberg TS, Chefer VI, Zapata A, Heidbreder CA. (2001) Modulation of the behavioral and neurochemical effects of psychostimulants by kappa-opioid receptor systems. Ann. N Y Acad. Sci. 937:50-73.

Shippenberg TS, Herz A. (1986) Differential effects of mu and kappa opioid systems on motivational processes. NIDA Res Monogr. 75:563-566.

Sidman M. (1956) Time discrimination and behavioral interaction in a free operant situation. Journal of Comparative and Physiological Psychology. 49(5):469-473.

Sinha R. (2001) How does stress increase risk of drug abuse and relapse? Psychopharmacology (Berl.). 158:343–359.

Skrabalova J, Drastichova Z, Novotny J. (2013) Morphine as a potential Oxidative Stress-Causing Agent. Mini-Reviews in Organic Chemistry. 10(4):367-372.

Swanson LW. (1982) The projections of the ventral tegmental area and adjacent regions: a combined fluorescent retrograde tracer and immunofluorescence study in the rat. Brain Res Bull. 9(1-6):321-53.

Tian L, Hires SA, Mao T, Huber D, Chiappe ME, Chalasani SH, Petreanu L, Akerboom J, McKinney SA, Schreiter ER, Bargmann CI, Jayaraman V, Svoboda K, Looger LL. (2009) Imaging neural activity in worms, flies and mice with improved GCaMP calcium indicators. Nature Methods. 6:875–881.

Tice DM, Bratslavsky E, Baumeister RF. (2001) Emotional distress regulation takes precedence over impulse control: if you feel bad, do it! J Pers Soc Psychol. 80(1):53-67.

Timár J, Gyarmati Z, Fürst Z. (2005) The development of tolerance to locomotor effects of morphine and the effect of various opioid receptor antagonists in rats chronically treated with morphine. Brain Research Bulletin. 64(5):417-424.

Tinbergen N. (1951) The study of instinct. New York: Oxford University Press.

Tomkins S. (1966) Psychological model of smoking behavior. Am J Public Health. 56:17–20.

Unsworth N, Engle RW. (2007) The nature of individual differences in working memory capacity: active maintenance in primary memory and controlled search from secondary memory. Psychol. Rev. 114:104–132.

Wall PM, Messier C. (2002) Infralimbic kappa opioid and muscarinic M1 receptor interactions in the concurrent modulation of anxiety and memory. Psychopharmacology. 160:233–244.

Walker BM, Kissler JL. (2013) Dissociable effects of kappa-opioid receptor activation on impulsive phenotypes in Wistar rats. Neuropsychopharmacology. 38:2278–2285.

Watson JB, Rayner R. (1920) Conditioned emotional reactions. Journal of Experimental Psychology, 3, 1–14.

Weerda R, Muehlhan M, Wolf OT, Thiel CM. (2010) Effects of acute psychosocial stress on working memory related brain activity in men. Hum. Brain Mapp. 31(9):1418-1429.

Wellman CL. (2001) Dendritic reorganization in pyramidal neurons in medial prefrontal cortex after chronic corticosterone administration. J Neurobiol. 49(3):245-53.

Wilbrecht L, Holtmaat A, Wright N, Fox K, Svoboda K. Structural plasticity underlies experience-dependent functional plasticity of cortical circuits. J Neurosci. 30(14):4927-32.

Wills TA, Shiffman S. (1985) Coping and substance abuse: a conceptual framework. In: Shiffman S, Wills TA (eds) Coping and substance use. Academic Press, Orlando, pp 3–24.

Wu H. (2013) Higher-Order Assemblies in a New Paradigm of Signal Transduction. Cell. 153(2):287-292.

Van't Veer A, Carlezon WA Jr. (2013) Role of kappa-opioid receptors in stress and anxiety-related behavior. Psychopharmacology (Berl). 229:435–452.

Vasterling JJ, Brailey K, Constans JI, Sutker PB. (1998) Attention and memory dysfunction in posttraumatic stress disorder. Neuropsychology. 12:125–133.

Volkow ND, Li TK. (2004) Drug addiction: the neurobiology of behaviour gone awry. Nat Rev Neurosci. 5:963–970.

Vyas A, Bernalb S, Chattarji S. (2003) Effects of chronic stress on dendritic arborization in the central and extended amygdala. Brain Res. 965:290–294.

Yan W-S, Li Y-H, Xiao L, Zhu N, Bechara A, Sui N. (2014) Working memory and affective decision-making in addiction: A neurocognitive comparison between heroin addicts, pathological gamblers and healthy controls. Drug and Alcohol Dependence. 134:194-200.

Yasuda R, Nimchinsky EA, Scheuss V, Pologruto TA, Oertner TG, Sabatini BL, Svoboda K. (2004) Imaging calcium concentration dynamics in small neuronal compartments. Sci STKE. 2004(219):pl5.

Yin HH. (2010) The sensorimotor striatum is necessary for serial order learning. JNeurosci 30:14719 –14723.

Zalocusky KA, Ramakrishnan C, Lerner TN, Davidson TJ, Knutson B, Deisseroth K. (2016) Nucleus accumbens D2R cells signal prior outcomes and control risky decision-making. Nature. 531(7596):642-646.

APPENDICIES

APPENDIX I – CAPSTONE THESIS

Optimizing methods of optical modulation and recording of neuronal activity *in vivo*

Bioengineering Capstone Thesis, University of Washington

Mackenzie Andrews
Chavkin Laboratory
Antony Abraham • Alyssa Taylor

TABLE OF CONTENTS

ABSTRACT

Motivation: Drug addiction is a highly prevalent disease with common potentiating risk factors including stress-exposure, anxiety, and depression. Several brain regions have been linked to drug seeking and abuse behaviors, however little is known about how these regions communicate to regulate those behaviors. In order to gain a better understanding of the networks driving addiction associated behaviors, there is a need for devices that can simultaneously record and modulate neural activity in multiple brain regions.

Contents: This report discusses neuropharmacological research in the field of drug abuse and the drawbacks of current investigational strategies. It then outlines the needs, specifications, and design constraints for developing a new research device and tracks the design process through to the development of a novel research device. Finally, it discusses the experimental design and preliminary computational results pertaining to the project.

Results: The deliverables of this project are (1) an optimized research device capable of dual-site recording and modulation of neuronal activity and (2) a proof-of-concept computational program that validates the ability to time-lock neural activity recording to behavioral events to offer insight to the cellular activity underlying reward-driven behaviors.

INTRODUCTION

BACKGROUND

Significance of Drug Abuse

Drug abuse is highly prevalent in the United States with approximately 25 million Americans (over 9 percent of the population) using an illicit drug in any given month (NIDA, 2015). Stress-exposure is a widely-recognized risk factor leading to compulsive drug abuse and relapse during abstinence (Goeders, 2002). Addiction potentiating stressors such as anxiety and major depressive disorders are the most common mental illnesses in the United States with anxiety affecting about 18% of adults in any given year (Kessler, 2005) and major depression inflicting 6.7% of all U.S. adults (Center for Behavioral Health Statistics and Quality, 2016). The relationship between stress and addiction is reciprocally compounding where stress exposure increases addiction risk and drug exposure increases the vulnerability to stress.

Neuropharmacology of Drug Abuse

Several brain regions have been implicated in drug seeking and abuse related behaviors. A pathway of particular interest is the mesolimbic dopamine system which mediates reward signaling. This pathway originates in a dopamine-rich nucleus called the ventral tegmental area (VTA). These dopaminergic neurons project primarily to the nucleus accumbens (NAc) which processes the reward signal. Other brain regions interact with this pathway such as the amygdala which is involved in reward valuation and is important in habit formation, craving, and withdrawal. Other regions have important functions that mediate drug seeking such as the hippocampus for memory, hypothalamus for regulating stress and motivation, and pre-frontal cortex (PFC) which is responsible for decision making and may play a role in impulsivity.

At a molecular level, studies on the relationship between stress and addiction have shown that stress-exposure causes the release of dynorphins, a class of endogenous opioid peptides. Dynorphins act primarily through the κ-opioid receptor (KOR) to potentiate the rewarding effects of drugs of abuse including cocaine, ethanol, and nicotine (Chavkin, 2014). Dynorphin is widely produced in the central nervous system (CNS) and regulates neurons that project to many of the

aforementioned brain regions including the VTA to regulate their excitability (Figure 1) (Chavkin, 2014).

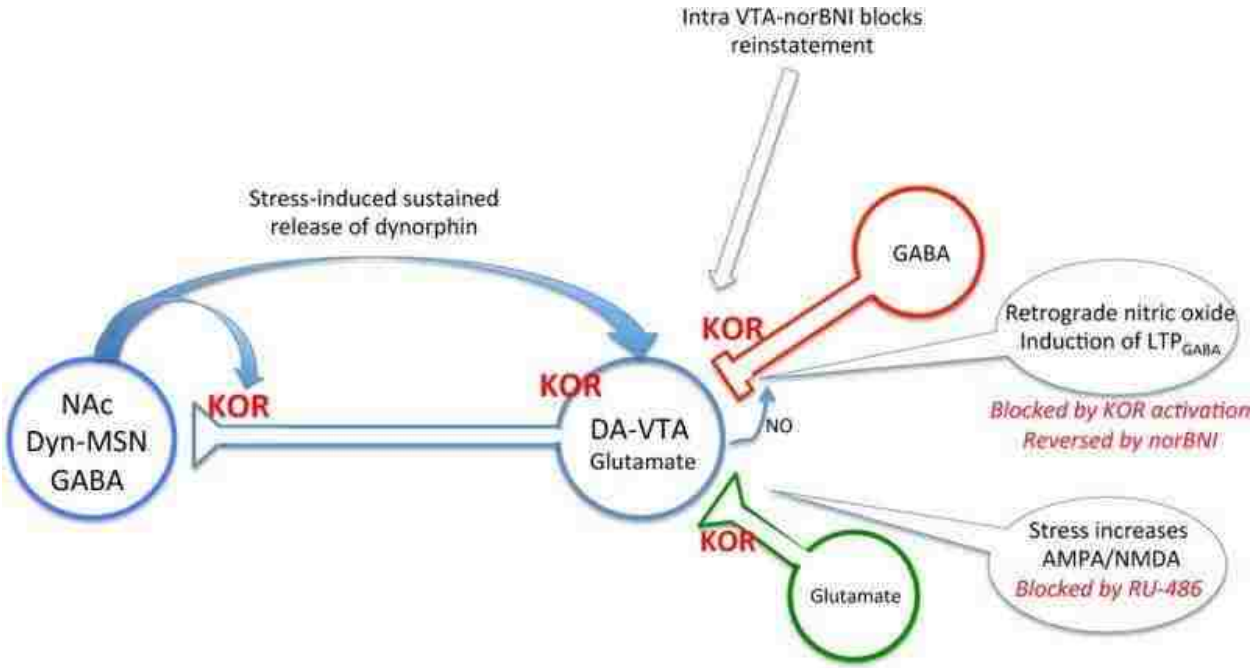

*Figure 1: **KOR in the Mesolimbic Dopamine Pathway –** Schematic showing KOR-mediated GABAergic and glutamatergic input to the VTA. The VTA has dopaminergic and glutamatergic projections to the nucleus accumbens (NAc) also mediated by KOR. These projections signal 'incentive salience' to the NAc, a primary driver of addictive behavior. The NAc also sends dynorphin containing afferents to the VTA presumably contributing to the reciprocal stress-addiction relationship.*

INVESTIGATIONAL TOOLS AND STRATEGIES

Current Methods for Investigating Neural Circuitry

Our current knowledge of the brain regions and pathways involved in drug seeking and abuse has come from an evolution and advancement of investigational tools and strategies. General investigational approaches include cell labelling, recording of neural activity, genetic modification, and artificial modulation of neural signals.

Neuronal Tracers

One of the earliest technologies developed to investigate neural connectivity was neuronal tracers. Neurons have existing machinery in place to transport materials between the cell body and axon terminals. Anterograde transport is the movement of materials towards axon terminals and retrograde transport is movement of materials to the cell body. Neuronal tracers

use the existing axonal transporters in neurons to transport labelled particles through the cell. Neuronal tracing studies have been important for the field because they allow for identification of neuronal pathways and functions (Oztas, 2003).

A typical tracing experiment will involve the injection of a labelled dye into a brain region of interest. This dye will be taken up by the neurons and transported along their axons. In retrograde tracing experiments, the dye will be taken up by the axons of the neurons that project to the region of interest and will be transported to label the location of the cell bodies of those neurons. In anterograde tracing experiments, the dye will be taken up by the cell bodies in the region of interest and transported to the axon terminals to show the regions where those neurons project.

Despite its powerful ability to elucidate brain connectivity, one downside of neuronal tracing is that it doesn't offer information about the activity of the neurons that are being labeled. In basic tracing experiments, weak neural connections will be labeled the same as strong neural connections, so it is difficult to parse out which regions and networks are important for driving animal behavior.

Electrophysiology

In order to investigate and record the activity of neurons, electrophysiological methods have been used. Electrophysiology is the study of the electrical properties of cells through the measurement of voltage changes or electrical current. In neuroscience, electrophysiology is used to measure the electrical activity of neurons. Electrophysiological methods can be used to measure activity from the scale of a single ion channel to broad field potentials of entire brain regions (Nicolelis, 2002).

In order to study how the activity of individual neurons and neural connections are driving behavior, *in vivo* electrophysiology is used. For *in vivo* electrophysiology recording, an electrode (typically a conductive wire or silicon probe) is inserted into the brain region of interest. The animal is then attached to an electrical tether so that the neural activity can be recorded and saved to a signal acquisition system. The animal is then able to move freely to conduct a behavioral test or for the researcher to observe the animal behavior.

In vivo electrophysiology has allowed researchers to learn how the activity of neurons drives or modulates the behavior of an animal. However, the downside to this type of recording method is that its difficult to tell what subtype of neurons is being recorded from and it does not offer a way to change or modify the neural activity in order to ask experimental questions.

Genetic Modification

In order to investigate how specific subtypes of neurons, receptors, or endogenous molecules drive behavior, genetic tools were developed to alter the expression or regulation of the variable in question. Genetic investigation involves germline modifications of early embryos. These approaches allow researchers to upregulate ('knock-ins'), downregulate ('knock-downs'), or delete ('knock-outs') specific genes (Lowenstein, 2001). These techniques allow researchers to investigate how changes in gene expression of a research target alter animal development and behavior.

Genetic modification is an extremely powerful tool in neuroscience because it elucidates the role of specific sub properties of neurons. However, a downside is that it does not allow for real time modulation of neural activity. When researching a disease such as addiction, where the functioning of the brain changes over the course of the disease, its important to be able to ask questions about how modulation of specific subtypes of neurons can modify animal behavior in real time.

Optogenetics

Fortunately, the combination of genetic techniques with light activated proteins has developed a tool for real-time, reversible modulation of neuronal activity. Optogenetics involves the insertion of a light activated protein, or 'opsin', into the cell membrane. When exposed to a light with a wavelength within the protein's activation range, the conformation of the protein will change, typically opening an ion channel. The opening of a channel will result in the flow of ions in or out of the cell, which will change the excitability profile of the cell. The first, and most common opsin used for optogenetic experiments is channelrhodopsin-2 (ChR2) (Buchen, 2010). When activated by blue light, ChR2 allows the flow of positive ions into the cell, depolarizing the membrane, increasing the cell's excitability.

The use of optogenetics *in vivo* has allowed researchers to investigate how the modulation of neural activity affects animal behavior in real-time. To do optogenetic investigation *in vivo*, an animal must be genetically modified or virally infected in order to express the opsin protein. Optogenetics allows sub-type specific expression of the opsin, i.e. the opsin can be differentially expressed in a specific type of neuron in a specific region of the brain. A light fiber is then implanted in the brain in the region of interest. The fiber is then connected to a laser which is controlled by the researcher to send pulses of light to the brain. The animal is able to move freely and perform behavioral tasks.

The real-time modulation of neural activity during behavioral tasks has elucidated how specific subtypes of neurons or regional neural activity affects behavior and brain function including memory consolidation (Rolls, 2011), impulsivity (Miyazaki, 2014), and drug seeking (Stefanik, 2012). However, optogenetics alone doesn't give researchers detailed information about the real-time activity of neurons, because there are no neural recording modalities in place.

Optrodes

In order to acquire neuronal activity measurements while maintaining the ability to artificially modulate the cellular activity, combinatorial methods must be used. Researchers have developed devices in which electrodes are implanted alongside an optical fiber, terming the name 'optrode'. A major benefit of optrodes is that optically stimulating a neuron will not produce any electrical artifacts, so the cells can be modulated in real-time while not disrupting the recorded signal.

Optrodes have traditionally been chronically implanted in rodents and used for *in vivo* modulation and recording of neural activity during behavioral tasks. Following is a brief review of some of the landmark studies and reviews that have led to the current state-of-the-art optrode devices.

The Evolution of the Optrode

Targeting and Readout Strategies for Fast Optical Neural Control In Vitro and In Vivo

In 2007, Gradinaru et al. wrote a review on the first strategies combining optogenetic modulation with electrophysiological recording. They described that optogenetics allow for the

targeting of cells in various ways such as viral mediated injection during development to incorporate the proteins into neural layers, injection with promoter fragments that only promote in specific subtypes of neurons or specific subcellular portions of the neurons, and also the simple projection of the photo stimulus. All of these targeting methods allow for more specified research from locations and subtypes of cells even to the subcellular level of a single neuron. The group showed examples of research that had been done that shows these methods were all quite effective in different ways (Figure 2).

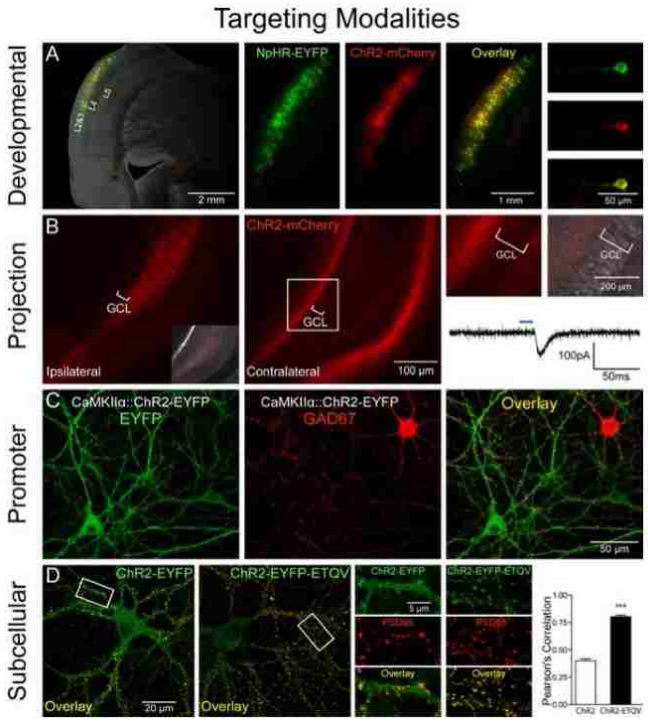

Figure 2: ***Optogenetic Targeting Techniques*** *- Immunohistochemical validation of a number of opsin targeting modalities including through breeding (A), via axonal projection (B), viral promoter targeting (C), and subcellular targeting (D). (Adapted from Gradinaru et al., 2007)*

The group described that the implementation of optically stimulated proteins can be paired with numerous methods of data recording. A major benefit being the fact that optically stimulating a neuron will not produce any electrical artifacts. It can be paired quite effectively with electrophysiological signal recording done via a simple electrode. Due to the high speed of these signals and effective recording, very high temporal resolution can be seen. Optical neural control can also be paired with other methods such as luminescent optical recording and long

term biochemical analysis. If the speed of acquisition and optical stimulation delivery is sufficient, luminescent optical recording can be used to determine the molecular and ionic information concerning neuronal and local circuit action potentials. The long term biochemical analysis on the other hand, may allow for the long-term tracking and validation of the effects optical neural control has on neurons and circuits. Through simple visual analysis however, larger scale changes can be seen through the use of optical neural control. Specifically, the behavior of conscious rodents can be altered via optogenetic stimulation. This could be seen in an experiment in which, only the right side of a rodent's brain was optogenetically stimulated while awake. This experiment resulted in the continued tendency for the rodent to turn left and walk in a counterclockwise spiral while stimulated (Figure 3).

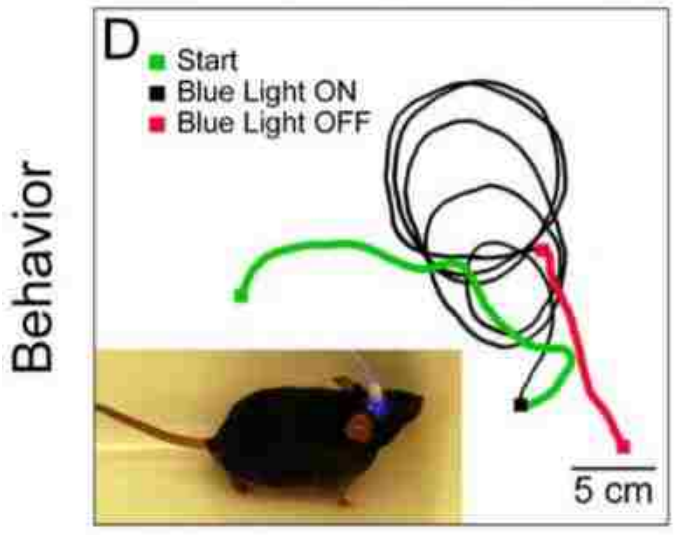

Figure 3: ***Optical Control of Behavior*** *– Walking behavior of a mouse before (green), during (black), and after (red) optical stimulation. Optical control of the right motor cortex caused the animal to walk in left hand circles. (Adapted from Gradinaru et al., 2007)*

Multi-array silicon probes with integrated optical fibers: light-assisted perturbation and recording of local neural circuits in the behaving animal

Once the idea of the optrode was exposed to the field, there was a major push to increase the spatial resolution of the devices. In order to achieve this aim, Royer et al., (2010) developed multi-array silicon probes equipped with micron-scale optical fibers for optogenetic probing of neural circuits. Figure 4 shows the general engineering design and images of the completed device. The group etched the optical fiber using hydrofluoric acid and then cured the fiber to the

silicon probe via UV curable epoxy (Figure 4A-B). The optical fiber equipped shanks were then arranged in either 4 or 8 shank probes (Figure 5). The shanks were designed such that the optical fiber could terminate between 300 micrometers before the tapering of the shank to 100 micrometers past the end of the shank, allowing for versatility in where the light modulation was occurring.

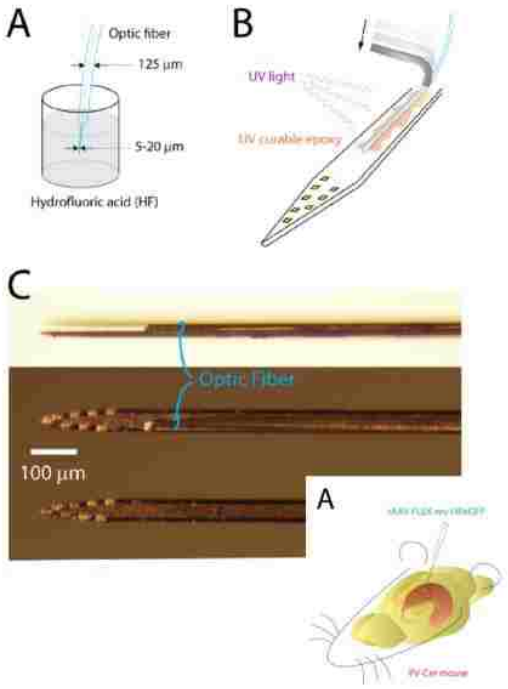

*Figure 4: **Silicon Array Schematic –** A-top) Etching of optical fiber via hydrofluoric acid; B) Securing of optical fiber to silicon array via UV curable epoxy; C) Images of completed silicon array; A-bottom) Schematic of implantation into a rodent. (Adapted from Royer et al., 2010)*

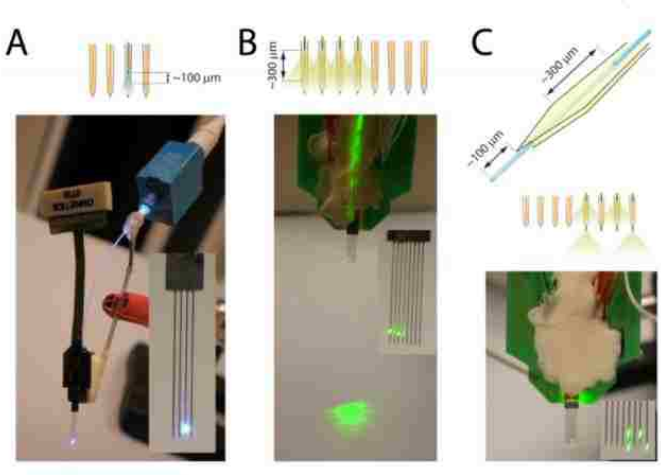

Figure 5: ***Multi-Shank Silicon Probe –*** *A) 4-shank probe with one light equipped channel; B) 8-shank silicon probe with 4 light equipped channels; C) 8-shank silicon probe with 4 light equipped channels of variable fiber optic termination location. (Adapted from Royer et al., 2010)*

Using these multi-shank silicon probes, the group injected a parvalbumin targeting halorhodopsin virus into the hippocampus of mice. By implanting a 4-shank probe with one light equipped shank into the halorhodopsin expressing animals, the group was able to show spatial specificity in neural modulation via the light equipped channel (Figure 6). The 8-shank probe was capable of recording from "50 to 140 well-clustered neurons," which was a significant increase in spatial resolution compared to previous methods. Using this technique, the group went on to show an example of a behavioral task in which light modulation was applied, selectively activating specific neurons during the task (Figure 7).

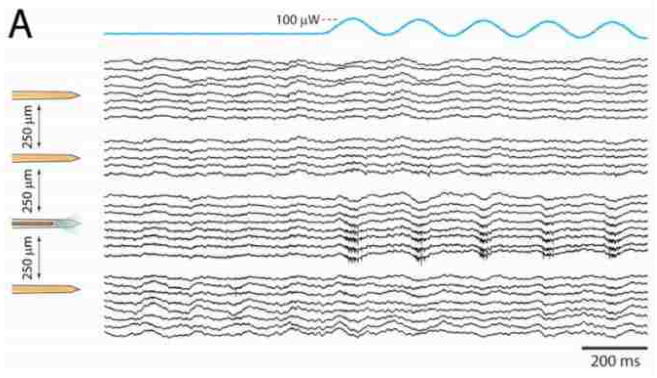

Figure 6: ***Light Modulation –*** *Sinusoidal light modulation on one channel of a 4-shank probe showing spatial specificity in neural activity modulation. (Adapted from Royer et al., 2010)*

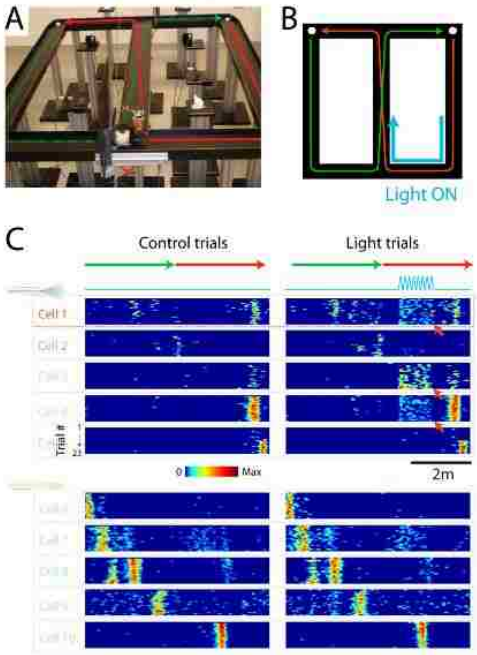

Figure 7: ***Neural Modulation During Behavior –*** *A) Behavioral set up; B) Light modulation during specific behavioral region; C) Recordings from 10 cells during behavior showing specificity inn responsiveness. (Adapted from Royer et al., 2010)*

The flexDrive: an ultra-light implant for optical control and highly parallel chronic recording of neuronal ensembles in freely moving

Chronic recording of neuronal ensembles is challenging, especially in behaving mice. The complexity of available drive mechanisms, implant weight tolerated by mice, and finding the precise positioning for high-quality recording, limits current methods to recordings from no more than 4–8 electrodes in a single target area. This becomes even more complicated when integrating optic fibers for optogenetic manipulations. Mice comfort and survivability imposes weight limit of about 4g, and even 2g when a combination of recording with behavioral phenotyping is desired (motion and videography needs). Previous studies bypassed the weight problem by offsetting the implant weight with a pulley system or attaching a helium-filled balloon to the implant, or by placing the animal on a headpost, which aren't ideal. Another method utilizes static electrode arrays to make large-scale recordings within a lightweight device. However, in case of inaccurate positioning or glial encapsulation, use of static electrodes prevents an easy change in position. There is a need for a low weight, with high number of individually movable electrodes, high placement stability, and independently adjustable optical fibers that would allow high-quality recording in behaving mice.

A highly miniaturized drive design that replaces the drive mechanisms found in current implants with a one-piece spring design was developed. “FlexDrive” (Figure 8) fits 16 individually movable tetrodes (up to 64 channels), can maintain stable recording conditions for months, integrates 2+ optic fibers, and only weighs ~2g (Figure 8B). Electrodes are positioned by an array of flexible polyimide guide tubes, while the electrode sits in a movable “shuttle tube” that can be laterally adjusted using screws (Figure 8A), where each screw turn corresponds to a 250µm lateral motion. This allows the user to adjust the position of individual electrode, even after surgery, which is important for several reasons like in the case of glial encapsulation.

An injection of adenovirus was used to deliver the gene for channelrhodopsin-2 (ChR2) to the SI region and the device was mounted, all in the same surgical procedure (Figure 9A). The authors were able to show recorded activity from relevant neurons in response to laser pulses (Figure 9B). In addition, a multi-site recording from both the SI region and the thalamic reticular nucleus (RTN) was achieved by adjusting individual electrodes, showing that the design can produce parallel recordings from 16 electrodes from two sites with the accuracy required to observe neurons in small, deep targets such as the TRN. To conclude, the flexDrive presents a straightforward method for obtaining stable and high-quality electrophysiological data from multiple target sites in awake, behaving mice. This permits researchers to make full use of the precision and specificity of optogenetic methods by directly probing the concerted function of neural circuits, rather than individual neurons.

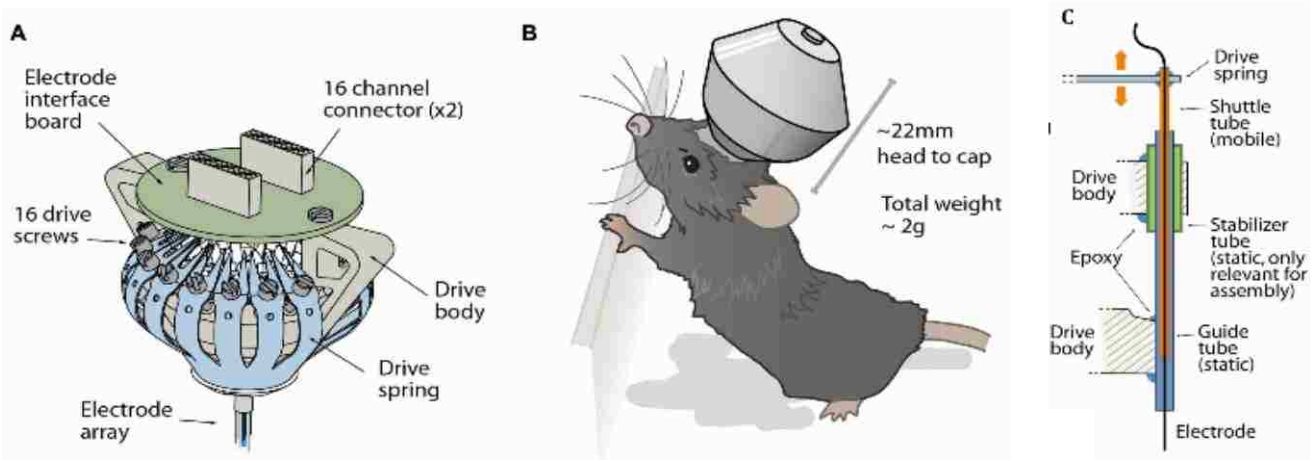

*Figure 8: **flexDrive Schematic** - A) Internal view of flexDrive featuring the interface board, the 16 adjusting screws, the polyimide drive body, and the drive spring; B) External view of the flexDrive mounted on a mouse’s head, weighing only ~2g and 22mm head to cap; C) Adjustable electrode mechanism. The mobile shuttle tube is adjustable by the drive spring (connected to the screw) and slides laterally inside the guiding tube to control the position of the electrode. (Adapted from Voigts et al., 2013)*

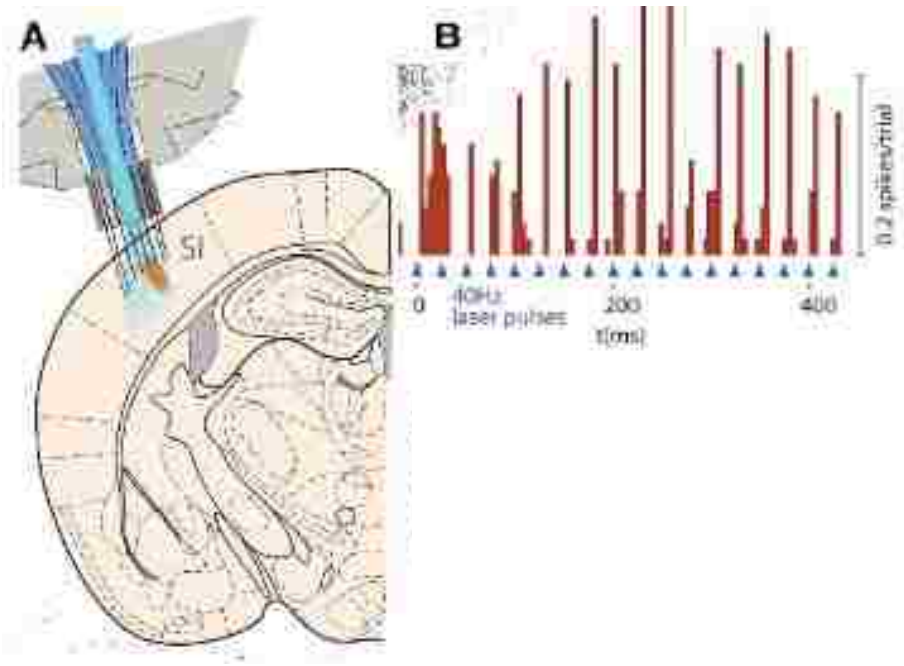

*Figure 9: **Optical Modulation with flexDrive** - A) Activation of neurons in S1 region with ChR2; B) Trace of an identified neuron on one of the tetrodes for one session. (Adapted from Voigts et al., 2013)*

Drawbacks of Current Designs

A number of advances have been made in the field of optogenetic modulation coupled with electrophysiological recording. The flexDrive currently offers the largest number of individually drivable recording channels and is the first device capable of stimulating and recording from more than one brain region simultaneously. However, the flexDrive limits the regions that it can record from to a lateral separation of no more than 1.5 mm.

The primary reason that no devices have the capability to record from multiple brain regions with high spatial separation is the limitations of the size and weight of the devices. As mentioned previously, the upper bound on weight for chronic implantation and behavioral recording in mice is approximately 2g. Current devices such as the flexDrive are already bordering on this weight limit and thus could not possibly be implanted in multiple regions. Even if the current devices weighed less, their sheer size is a limiting factor for multi-site implantation on a mouse skull.

STATEMENT OF PROBLEM AND NEED

While the field of drug abuse research has identified important brain centers, pathways, molecular agents, and targets responsible for mediating addictive behaviors, there is a gap in

knowledge about how these regions actively communicate in awake and behaving animals. In order to develop new treatments for addiction and related mental illnesses, we must understand how neuronal activity, at a multi-region, circuit-wide level regulates specific behaviors such as drug seeking.

Recording from a freely behaving animal, requires devices that are small enough to fit on an animal's head and light enough to be carried by the animal. In addition to the size constraints, the devices must be able to modulate and record neuronal activity with high spatiotemporal resolution. Current devices that meet these requirements are only capable of recording from one brain region at a time or are limited to simultaneous recordings from regions with low spatial separation.

The head mount assemblies must include an electrode interface board (EIB), electrode fibers, an optical fiber, and a driving mechanism to adjust fiber position. In addition, the fiber driving mechanisms must be easily accessible by the researcher to adjust fiber position between recording sessions.

In order to achieve a multi-site recording device, significant design modification must be made to reduce the weight and size of the device while maintaining high spatiotemporal resolution and fiber driving capabilities.

DESIGN PROCESS

CURRENT MODEL DEVICE – ZWEIFELTRODE

The current optrode device used in the University of Washington Pharmacology Department was designed by Yong Jo in the lab of Larry Zweifel, termed 'Zweifeltrode'. This device was the basis for all design modifications, optimizations, and validation testing described in this report.

Device Description

The Zweifeltrode (Figure 10) is a single-site optrode consisting of a single optical fiber, and four tetrode bundles resulting in 16 electrode wires. The electrodes interface to an EIB containing a female connector for the data acquisition system. The optical fiber terminates in an optic ferrule for connection with the laser driving system. The device contains a screw driving mechanism to drive the optic and electrical fibers simultaneously into the brain. The device is manufactured with 3D printed parts and assembled entirely in-house.

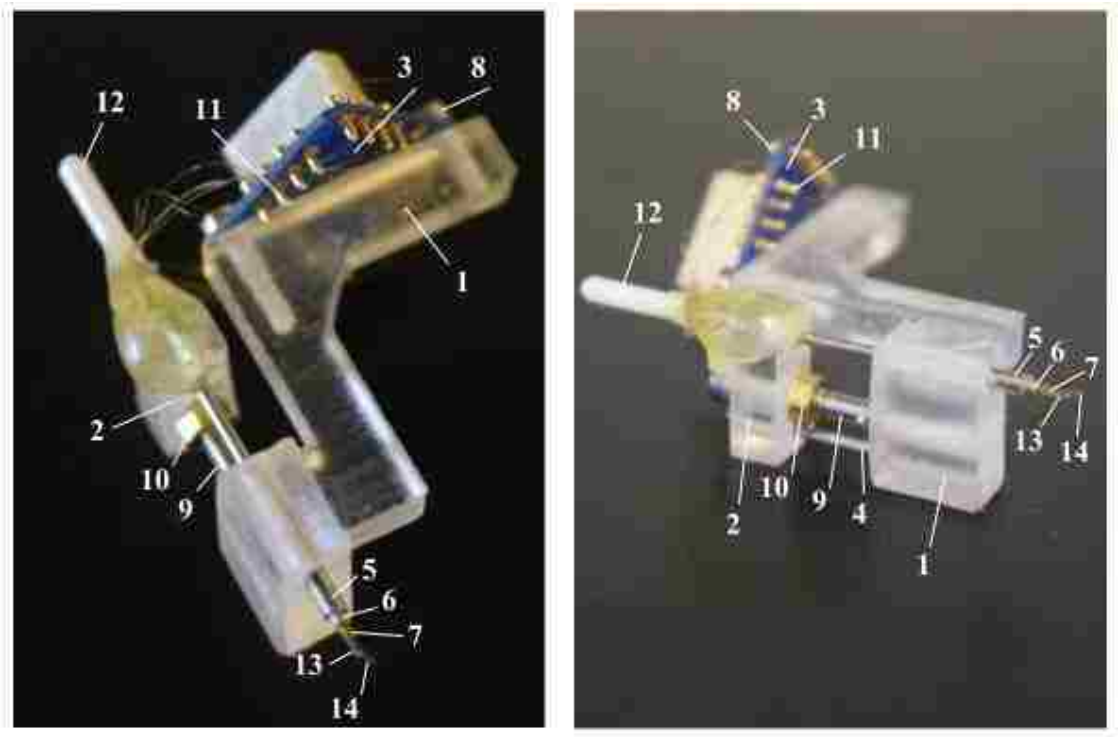

*Figure 10: **Images of Zweifeltrode –** Two perspective images of the current model optrode used by the University of Washington Pharmacology Department. This single-site optrode was design by Yong Jo of the Zweifel Lab. Numbered part labels correspond to the part reference numbers in Table 2: Zweifeltrode Parts List.*

Device Specifications

The following is a table of relevant specifications for the current model device. These are the specifications that will be compared to the new device for validation of device optimization.

TABLE 1: ZWEIFELTRODE SPECIFICATIONS

Description	Metric
Size	
Height	15 mm
Width	18 mm
Length	14 mm
Weight	1.9 g
Drive Mechanism	
Drive Range	3.5 mm
Drop/turn	.212 mm
Recording Capabilities	
Number of Sites	1
Electrodes/site	16
Optic Fibers/site	1

Parts List

The following is a parts list for the current model device. All of the parts have a reference number corresponding the numbered parts in the device image, (Figure 10).

TABLE 2: ZWEIFELTRODE PARTS LIST

Part Description	Part Spec.	Source	Catalogue #	Ref. #
Drive Main	Yong CAD, ABS Plastic	Yong So	N/A	1
Drive Small	Yong CAD, ABS Plastic	Yong So	N/A	2
EIB	16-channel, 1x1x0.6cm, 200mg	Neuralynx	E1676292	3
Support Post	23G, 304 stainless steel, 15mm	SmallParts	HTX-23R-12-10	4
Outer Tubing	19G, 304 stainless steel, 7mm	SmallParts	HTX-19R-12-10	5
Inner Tubing	23G, 304 stainless steel, 16mm	SmallParts	HTX-23R-12-10	6

Polyamide Tubing	0.0137in, polyamide tubing	MicroLumen	BA863484	7
EIB Screws	Any Fit Acceptable	N/A	N/A	8
Drive Screw	0-80, 1 cm, stainless steel hex screw	N/A	N/A	9
Drive Screw Nut	0-80 Hex nut	N/A	N/A	10
Gold Pins	0.015-0.025in, gold plated	Neuralynx	Large EIB Pins	11
Optic Ferrule	1.25mm, 230um bore, Ceramic Zirc.	Prec. Fiber Prod.	MM-FER2007C	12
Optic Fiber	Silica, 0.22 NA, Ø200 µm Core	Thorlabs	FG200LEA	13
Tetrode Wire	.00099in, Tungsten 99.95%	CA Fine Wire Co	By Quote Only	14
Ground Wire	32G, Insulated Stainless Steel	Unknown	Unknown	15*
Ground Screw	Any Stainless Steel Screw Accept.	N/A	N/A	16*

*Not shown in device image

Assembly Protocol

The following is the assembly protocol for the current model device. The protocol is based on full in-house assembly with the tools available in the Zweifel Lab work bench space.

HARDWARE ASSEMBLY

I. Prepare metal cannula using dremel
    a. Cut two pieces of 23G tubing at 16mm (inner tubing) and 15mm (support post)
        i. Sand down
    b. Cut one piece of 19G tubing at 7 mm (outer tubing)
        i. Sand down
II. Prepare holes in Drive Main
    a. Thread middle hole with 0.08in screw driver from top
    b. Drill 2 holes for EIB board on top post with hand drill
III. Mount EIB with EIB screws
    a. '2' next to vertical post on Drive Main
IV. Mount tubing
    a. Place inner tubing in bridge hole on Drive Small from bottom
        i. Superglue from bottom
    b. Place support post in lateral hole on Drive Main from top
        i. Superglue from top
    c. Place outer tubing in medial hole on Drive Main from bottom
        i. Superglue from bottom
V. Prepare drive screw
    a. Place drive screw through center hole of Drive Small from top
    b. Glue drive screw nut
        i. Screw drive screw nut onto drive screw from bottom
        ii. Put superglue on drive screw close to bottom of Drive Small

iii. Screw drive screw nut into superglue

VI. Assemble Drive Small to Drive Main
- a. Place inner tubing into outer tubing from top
- b. Place support post into outer hole on Drive Small from bottom
- c. Screw drive screw into threading of middle hole on Drive Main
  - i. Ensure alignment and easy driving

VII. Mount polyamide tubing
- a. Thread polyamide tubing through inner tubing
  - i. Cut with 1mm clearance on both ends
  - ii. Superglue from top

TETRODE PREPARATION

I. Prepare tetrode wire
- a. Cut to 14 in (length of diagonal of standard 8.5x11in paper)
  - i. Ensure no kinks
- b. Fold in half once
  - i. Cut at loop
- c. Fold in half again
  - i. DO NOT cut
- d. Twist ends

II. Use tetrode spinner
- a. Mount wires
  - i. Place twisted ends in clamp of tetrode spinner
  - ii. Place loop on ring stand post
    1. Ensure wire is taut but not suspending clamp
- b. Spin wires
  - i. 35 Forward, 27 Reverse
    1. Adjust as necessary
- c. Heat Gun
  - i. 3 passes on high: Up; Down; Up

OPTIC FIBER PREPARATION

I. Cut optic fiber
- a. Use fiber stripper to remove insulation
- b. Measure length for brain region of interest
- c. Use diamond blade to score fiber
- d. Brake by bending

II. Polish fiber
- a. Place fiber(s) in holding rod of appropriate length
- b. Polish one end on polishing paper in ascending grit
  - i. Start with least perfect end
  - ii. Polish until no imperfections

III. Mount in ferrule
- a. Place non-polished end of fiber into ferrule
  - i. From open end to tapered end

b. Secure with epoxy
    i. Mix 2-part optic fiber epoxy
    ii. Place epoxy in open end around fiber
    iii. Cure with heat gun until dark brown

IV. Polish Fiber
a. Polish ferrule end on polishing paper in ascending grit
    i. Polish until no imperfections

FIBER MOUNTING

I. Thread tetrodes
a. Thread 4 tetrodes through polyamide tubing from top
b. Thread tetrodes to EIB
    i. Cut loop in tetrode
    ii. Thread wires through 4 holes on one side of EIB from bottom
c. Secure with gold pins from top
    i. Clamp with plyers

II. Thread optic fiber
a. Thread through polyamide tubing from top
    i. Ensure 1mm clearance at bottom end of polyamide

III. Cut tetrode wires
a. Trim wires from top
b. Cut wires at angle on bottom with 0.5-1mm clearance from bottom of optic fiber

IV. Epoxy optic fiber
a. Mix 2-part epoxy
b. Secure optic fiber with cone shaped epoxy mount from top

GROUND WIRE PREPARATION

I. Cut ground wire
a. Cut to 2in
b. Strip 0.5in insulation from both ends

II. Solder ground wire to ground screw
a. Wrap end of wire around top of screw threads
b. Solder to screw

III. Mount to EIB
a. Place end of wire through ground hole ‘G’ from bottom
b. Secure with gold pin from top

MULTI-SITE DESIGN CONSIDERATIONS

Using the Zweifeltrode as the model device, design iterations were made to design an optimized device with multi-site implantation capability. Throughout the design process, a number of device needs and design constraints were considered in order to develop a viable final

product. The following are the device requirements and additional considerations that were used throughout the design process.

Device Requirements

The requirements used in designing the new device were based on a number of factors. The first factor to consider is the needs of the parties that will be affected by the device design process. The stakeholders for this device are the research team (Chavkin Lab), the research organism (mice), the regulatory body that will approve the use of the device (The Institutional Animal Care and Use Committee, IACUC), and the financial stakeholders or parties funding the research. Each stakeholder has a different set of needs, however many of the needs between stakeholders overlap. Following is a summary of the primary needs of each of the stakeholders considered.

Design Needs

TABLE 3: DESIGN NEEDS

#	Need	Description
Stakeholder I – Chavkin Lab		
1	Answers Research Aim	Must answer the questions motivating the design process
2	Promotes Naturalistic Behav.	Allows animal to move freely to perform behav. tasks
3	Reproducible	Must be able to make multiple devices to replicate results
4	Low Cost	Comparable in cost to current model device
5	Easy to Use	Comparable ease of use as current model device
Stakeholder II – Mice		
2	Promotes Naturalistic Behav.	Allows animal to move freely to perform behav. tasks
6	Minimally Harmful	Does not cause excessive discomfort or harm to animal
Stakeholder III – IACUC		
6	Minimally Harmful	Does not cause excessive discomfort or harm to animal
7	Follows Mouse Implant Prot.	Within protocol requirements for implantation in mice
Stakeholder IV – Funding Parties		
1	Answers Research Aim	Must answer the questions motivating the design process

3	Reproducible	Must be able to make multiple devices to replicate results
4	Low Cost	Comparable in cost to current model device
8	Time Efficient	Can be completed within time frame of grants

The Chavkin Lab's reason for holding stake in the device is to answer the research questions posed. In order to meet those research aims, the device must produce novel data while promoting naturalistic behavior so that the animal can perform behavioral tasks. In addition, all results developed by the device must be reproducible, thus the device itself must be reproducible. The Chavkin Lab also needs the device to be low cost in order to be able to fund the project. Finally, the device must be easy for the researchers to use, i.e. it should not require any highly specialized training or advanced skill to use.

The mice are the most intimately affected stakeholder in the development of the device. The device must be minimally harmful to the animals and thus should not cause severe discomfort or unnecessary harm (e.g. infection or irritation). In addition, in order for the mice to perform the required behavioral tasks and survive independently, the device must allow the mice to behave naturalistically and must not inhibit independent movement, eating, and drinking.

The regulatory body, IACUC, is primarily concerned with the reasonable care and use of the animals for research. Thus, IACUC also requires that the device is minimally harmful and thus does not cause excessive discomfort to the animal. IACUC also requires that the device be within the size and weight constraints for implantation.

Finally, the parties funding the design of this device are the financial stakeholders. As financial stakeholders, the funding parties require that the device is low cost. In addition to being low cost, the device must effectively answer a novel research aim. Therefore, the funding bodies also require that the device produce reproducible, novel data. Finally, the funding bodies have time limits on their funding, thus the device must be made within the time requirements of the grants.

Needs-Metric Translation

In order to convert the stakeholder needs into evaluable design requirements, the needs had to be translated into design specifications. Each design specification must be accompanied

by a metric so that the final design specifications can be evaluated to ensure that the device design meets the stakeholder needs.

TABLE 4: NEEDS-METRIC

			Specification											
			Biocompatable	Aseptic	Accessible Driving Mech.	Light Weight	Small Size	Multi-site Recording	Appropriate Drive Range	Minimal Drops/Turn	Multiple Electrodes/Site	Incorporate Optic Fiber	Low Production Cost	Low Production Time
		Metric	MP	MP	T/F	g	cm^3	#	mm	mm	#	#	$	Day
Need	1	Answers Aim						**x**	**x**	**x**	**x**	**x**		
	2	Prom.Nat. Behav.				**x**	**x**							
	3	Reproducible											**x**	**x**
	4	Low Cost											**x**	
	5	Easy to Use			**x**									
	6	Min. Harmful	**x**	**x**		**x**	**x**							
	7	Follows Protocol	**x**	**x**		**x**	**x**							
	8	Time Efficient												**x**

MP = Material Properties, T/F = True or False

Design Constraints

In order to design the device with specifications that meet the needs described, the specifications had to be constrained to measurable values. In order to place values on the design constraints, a number of factors had to be considered. The first factor is the model organism. Using a mouse as the test organism places strict size and weight constraints on the device. Using a mouse also determined the appropriate drive range as the drive range must equate to the size of the brain region being recorded from. The next factors to be considered are the cost and time constraints placed on the project by the funding agencies. These factors are determined by the

terms of the grant. Following is a table displaying the design constraints determined by considering all of the necessary factors.

TABLE 5: DESIGN CONSTRAINTS

Specification	**Constraint**
Material Properties	
Biocompatible	Non-degradable plastics, 304 Stainless Steel
Aseptic	Sanitizable material
Size	**Per Site, Assuming 2 Sites**
Height	< 15 mm
Width	< 3 mm
Length	< 10 mm
Weight	< 1 g
Drive Mechanism	
Accessibility	By Hand
Drive Range	> 2 mm
Drop/turn	< 0.25 mm
Recording Capabilities	
Number of Sites	≥ 2
Electrodes/site	≥ 8
Optic Fibers/site	1

The material requirements are based on the definitions of biocompatible and aseptic materials. For the purpose of this device design, all materials used were the same as the materials in the model device. The use of ABS for 3D printing, polyamide tubing, and 304 surgical stainless steel from medical materials suppliers ensured that the materials were within the design criteria.

The size constraints were determined base on the model organism. The use of mice limits the weight of chronic skull implantation for behavioral assays to 2 grams (Voigts, 2013). In order to be implanted in two sites simultaneously, the device must be less than half the weight

constraint, therefore the device must be less than 1 gram. The length and width of the device are constrained by the size of the animal's skull and the regions of interest. Using mouse strain C57BL/6, the implantable skull area is 11.6 x 5.3 mm (Kawakami, 2008). This constrains the length of the device to approximately 10 mm. For the purpose of the design process, we are assuming recording from the PFC and VTA. The stereotaxic coordinates of the PFC are P: +1.8, L: ±0.3, V: 2.25 → 3.25 mm from Bregma; the stereotaxic coordinates of the VTA are P: -3.1, L: ±0.5, V: 4.3 mm from Bregma (Paxinos, 2012). The total separation between the implant coordinates for the VTA and PFC is 4.9 mm. To maintain accessibility, 2 mm must remain between the implants. This constrains the width of the device to 3 mm. The height constraint is based on the height of the current model device.

The range of the drive mechanism is based on the dorsoventral length of the brain regions of interest. The PFC is the longest region of interest with a dorsoventral length of 1mm. Assuming the fibers are implanted 0.5 mm above the region and are driven until they are 0.5 mm below the region, the total drive range must be at least 2 mm. In order to get at least 3 recording sessions per region in each animal, the maximal fiber drop per turn of the drive mechanism must not exceed 0.25 mm.

The electrode fiber count constraint was determined by the assumption that the fibers are bundled into tetrodes. In order to resolve spatial information, at least two tetrodes must be implanted per region thus requiring that each region have at least 8 electrode fibers.

Additional Considerations

In addition to the physical design parameters, additional considerations were made during the device and experimental design of this project. The following recognizes some potential concerns from an ethical, regulatory, and economic perspective and overviews the considerations made as a result.

Ethical

From a socio-ethical standpoint, the project was potentially controversial on two fronts. First, the involvement of animal research made the project prone to backlash from animal rights activist groups such as People for the Ethical Treatment of Animals (PETA). In order to protect the research from interference by said groups, all work strictly adhered to the IACUC protocols

for the proper treatment pf research animals and animal wellbeing was made a priority. However, being aware that many animal rights activists are entirely against all forms of animal research, privacy measures were taken to ensure that only approved persons had contact with animals.

Second, since the topic of research was drug abuse, the social stigma around addiction as a medical and societal disease makes research in the field a sensitive and controversial subject. In response to critique, outreach efforts were made to inform the public through press coverage and Washington State legislature through attending legislative sessions about the milestones in addiction research and potential impact the findings have on society.

Regulatory

From a regulatory standpoint, the project involves animal research, so all of the work had to follow and be approved by IACUC regulations (IACUC Guidebook, 2002). The project also involved the use of Schedule I drugs such as cocaine so all procedures had to adhere to the Drug Enforcement Administration (DEA) guidelines. Regular inspections by both of these regulatory agencies were mandated.

Economic

The major constraint for this project at the onset was funding limitations contingent on the grants available to the lab and the cost of the other projects being conducted by the lab. The major funding sources for the lab come from the National Institute of Drug Abuse (NIDA) and the National Institute of Mental Health (NIMH) which are both branches of the National Institute of Health (NIH). The current grants available to the Chavkin Lab have specific research aims associated with the funding and thus this project had to uphold to those aims. In addition to those funding sources, this project was funded by the Levinson Emerging Scholar Award which allowed a spending budget of $3,500.

DESIGN ITERATIONS

By taking the design needs, specifications, and constraints into consideration, the optrode device was redesigned over a number of iterations. The majority of the device optimizations were made by redesigning the driving mechanism. The iterations resulted in two landmark designs: an intermediary prototype and the final design to-date.

Prototype I – Coaxial Driving Mechanism

Taking in to consideration the needs and constraints of the final design, the most obvious component to optimize was the driving mechanism. The model driving mechanism consisted of 3 axle rods: the support post, the drive screw, and the fiber cannula. In order to consolidate the number of rods, the first design used a coaxial driving mechanism where the fiber cannula and drive screw were conserved in the same axle.

The coaxial driving mechanism consisted of 3 nested rods (Figure 11). The inner most rod housed the fibers and was externally threaded (Figure 11B-C, Blue). The center rod was internally threaded to the inner rod (Figure 11B-C, Red). The outer most rod was a guide shaft for the inner rods and would be mounted extracranially (Figure 11B-C, Green). The fibers would be driven by rotating the center rod which would drop the inner most rod (the fiber cannula) into the brain via the screw threads. In order to ensure that the fibers would not rotate relative to the brain, a torsional lock held the inner most rod rotationally stationary relative to the outer rod.

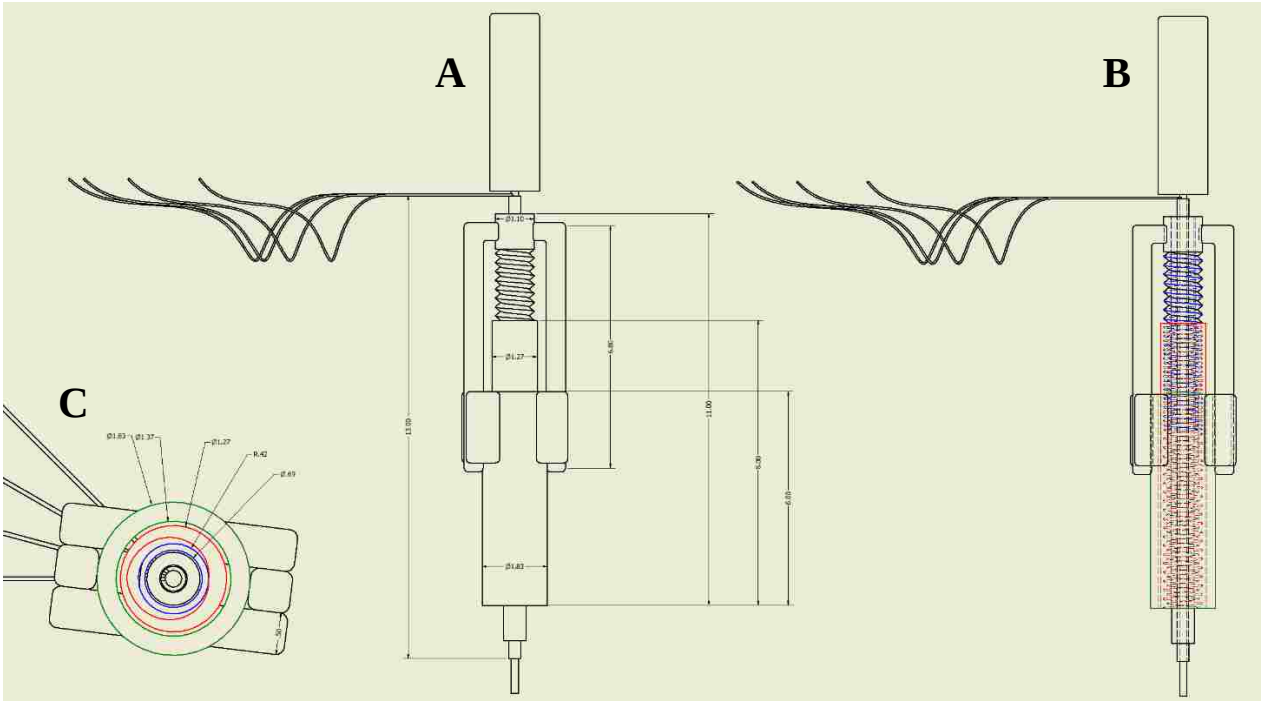

Figure 11: ***Prototype I Sketch*** *– A) CAD sketch of full prototype design including dimension labels. B) Transparent sketch with nested rods labeled by color. Blue: Inner rod housing fibers with external threading, Red: Center rod to drive inner rod with internal threading, Green: Outer rod acting as a guide shaft for drive mechanism. C) Ventral view of device with color labels corresponding to labels in B and rod diameter dimensions labeled.*

The design was projected to be made with 304 stainless steel tubing. The inner rod would be made of 20 gauge hypodermic tubing externally threaded with a 1.10 UNM die. The center rod would be made of 18 gauge hypodermic tubing internally threaded with a 1.10 UNM tap. The outer rod would be made of 16 gauge hypodermic tubing. The projected materials, dimensions, weight, and drive specifications all met the needs and constraints of the device (Table 6).

A micro-machinist (Kevin Willis, Verus Design & Prototype) was hired to manufacture the prototype. The materials and necessary tooling was ordered to manufacture the device. Unfortunately, the machining proved to be infeasible due to the thread size and brittleness of the needle stock. Therefore, manufacturing of Prototype I was abandoned (Figure 12).

TABLE 6: PROJECTED SPECIFICATIONS (PROTOTYPE I)

Specification	Constraint	Projected
Material Properties		
Biocompatible	Non-degradable plastics, 304 Stainless Steel	**Polyamide, 304 Stainless Steel**
Aseptic	Sanitizable material	**UV and EtOH sanitizable**
Size	**Per Site, Assuming 2 Sites**	**Per Site**
Height	< 15 mm	**13 mm**
Width	< 3 mm	**1.83 mm**
Length	< 10 mm	**2.83 mm**
Weight	< 1 g	**0.4 g**
Drive Mechanism		
Accessibility	By Hand	**By Hand**
Drive Range	> 2 mm	**3 mm**
Drop/turn	< 0.25 mm	**0.212 mm**
Recording Capabilities		
Number of Sites	≥ 2	**2**
Electrodes/site	≥ 8	**8**
Optic Fibers/site	1	**1**

*Figure 12: **Infeasible Manufacturing** – Manufacturing of Prototype I proved infeasible due to the thread size and material properties of 304 stainless steel hypodermic needle stock. Manufacturing of Prototype I was abandoned.*

Prototype II – Coaxial with Driving Nut

After abandoning manufacturing of Prototype I, the design was reiterated to account for feasibility of manufacturing. Through collaboration with the manufacturer, a new design was created (Figure 13). The new design adapted the coaxial drive mechanism to use a drive nut instead of an internally threaded rod. The fiber cannula remained as an externally threaded rod, the outer rod was conserved as a guide shaft. The torsional lock was also enlarged to ensure manufacturing feasibility.

The design was approved by the micro-machinist (Kevin Willis, Verus Design & Prototype) and was successfully manufactured (Figure 14). The torsional lock guide was 3D printed using ABS plastic. The torsional lock shafts were cut from 304 stainless steel. The fiber cannula was cut from a 0-80 thread stainless steel screw and a 0.027 inch diameter hole was bored from the center of the screw. The drive nut used was a 0.125 flats stainless steel hex nut. The outer rod was cut from 15 gauge, thin-walled, stainless steel needle stock. The final materials, dimensions, weight, and drive specifications all met the needs and constraints of the device (Table 7).

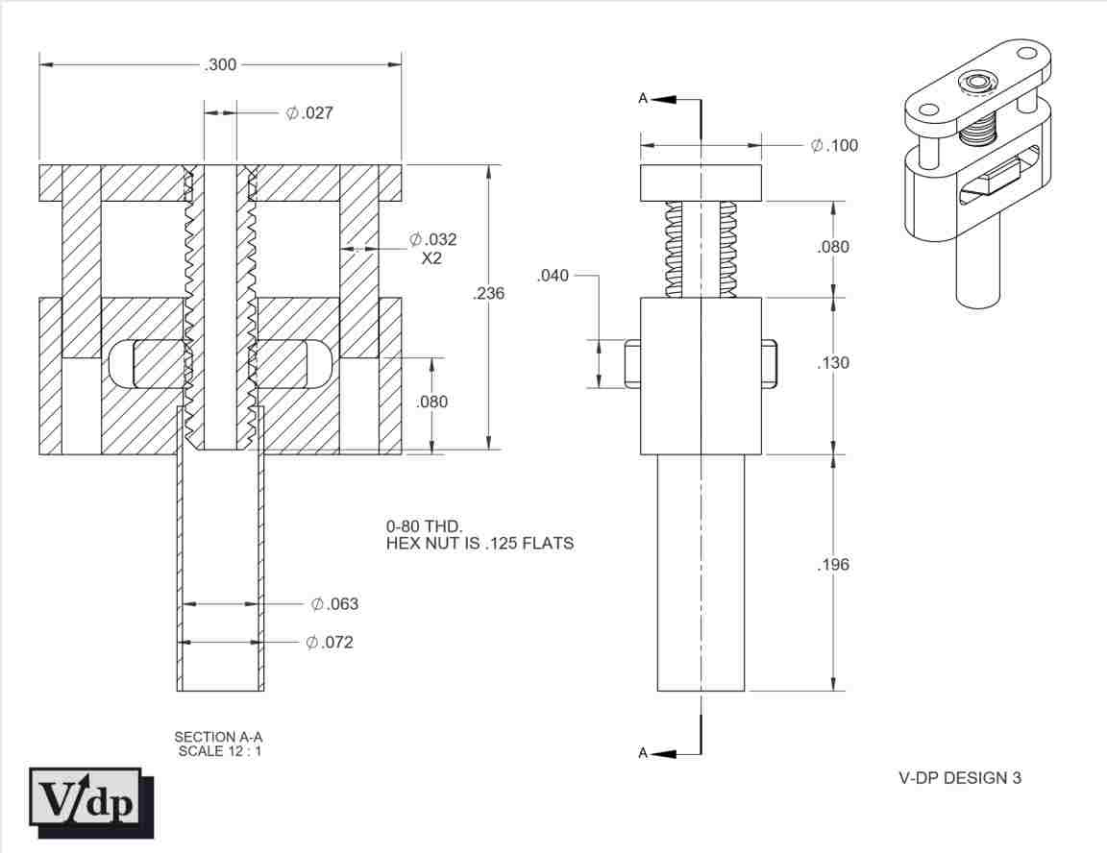

Figure 13: ***Protoype II Sketch –*** *Manufacturer approved sketch of prototype II including dimension labels, thread size, and drive nut specifications. Left: Slice of device through sagittal plane showing threading and torsional lock dynamics, Right: Side view of device showing slender dimensions, Upper-Right: Schematic sketch of prototype design.*

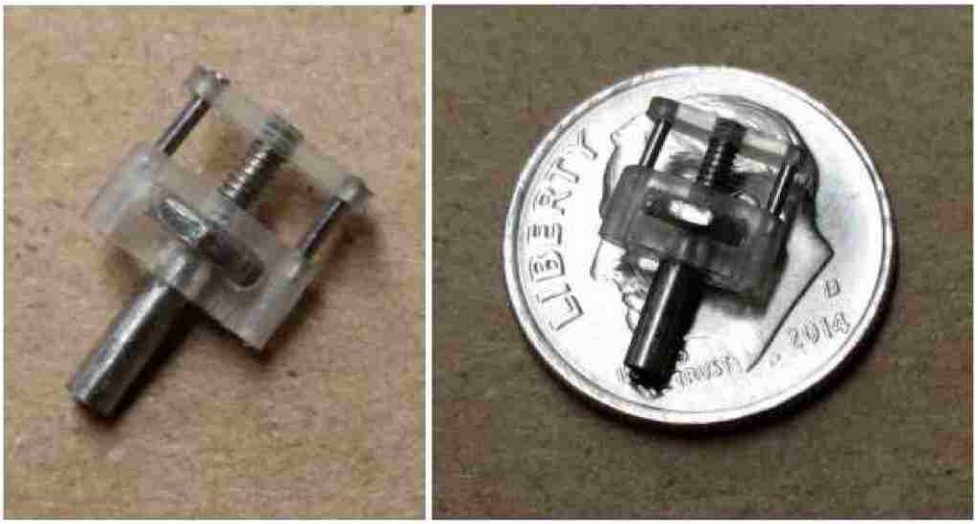

Figure 14: ***Manufactured Prototype II –*** *Images of the final manufactured product of prototype design II. The right image shows the device to scale on a US dime coin.*

Achieved Device Specifications

TABLE 7: ACHIEVED SPECIFICATIONS (PROTOTYPE II)

Specification	Constraint	Achieved
Material Properties		
Biocompatible	Non-degradable plastics, 304 Stainless Steel	**ABS, Polyamide, 304 Stainless Steel**
Aseptic	Sanitizable material	**UV and EtOH sanitizable**
Size	**Per Site, Assuming 2 Sites**	**Per Site**
Height	< 15 mm	**12 mm**
Width	< 3 mm	**2.5 mm**
Length	< 10 mm	**7.6 mm**
Weight	< 1 g	**0.7 g**
Drive Mechanism		
Accessibility	By Hand	**By Hand**
Drive Range	> 2 mm	**3 mm**
Drop/turn	< 0.25 mm	**0.212 mm**
Recording Capabilities		
Number of Sites	≥ 2	**2**
Electrodes/site	≥ 8	**8**
Optic Fibers/site	1	**1**

DEVICE VALIDATIONS

In order to ensure that the new design was a valid optimization of the previous device, a number of validation tests must be performed. The first validation is a comparison between the old design and the new design to ensure that the necessary optimizations were achieved while maintaining the device requirements. When comparing the new design to the previous design (Table 8), all required device components were maintained while significant optimizations were made to the device size (Figure 15). The new device only had decreased evaluations for the drive range and the number of electrodes per site. The drive range is still within the device constraints

and was understandably reduced because the drive mechanism was the primary component of optimization. The number of electrodes per site is restricted by the number of pins in the EIB divided by the number of recording sites. This number could easily be increased by using a larger EIB, however this would add to the total weight and size of the device.

TABLE 8: OPTIMIZATION COMPARISON

Specification	Previous Design	New Design	Optimization (%)
Material Properties			
Biocompatible	ABS, Polyamide, 304 Stainless Steel	ABS, Polyamide, 304 Stainless Steel	**Maintained**
Aseptic	UV and EtOH sanitizable	UV and EtOH sanitizable	**Maintained**
Size	Per Site	Per Site	
Height	15 mm	12 mm	**- 3 mm (20%)**
Width	18 mm	2.5 mm	**- 15.5 mm (86%)**
Length	14 mm	7.6 mm	**- 6.4 mm (46%)**
Weight	1.9 g	0.7 g	**-1.2 g (63%)**
Drive Mechanism			
Accessibility	By Screw Driver	By Hand	**No tool req.**
Drive Range	3.5 mm	3 mm	**- 0.5 mm (14%)**
Drop/turn	.212 mm	0.212 mm	**Maintained**
Recording Capabilities			
Number of Sites	1	2	**+ 1 (100%)**
Electrodes/site	16	8	**- 8 (50%)**
Optic Fibers/site	1	1	**Maintained**

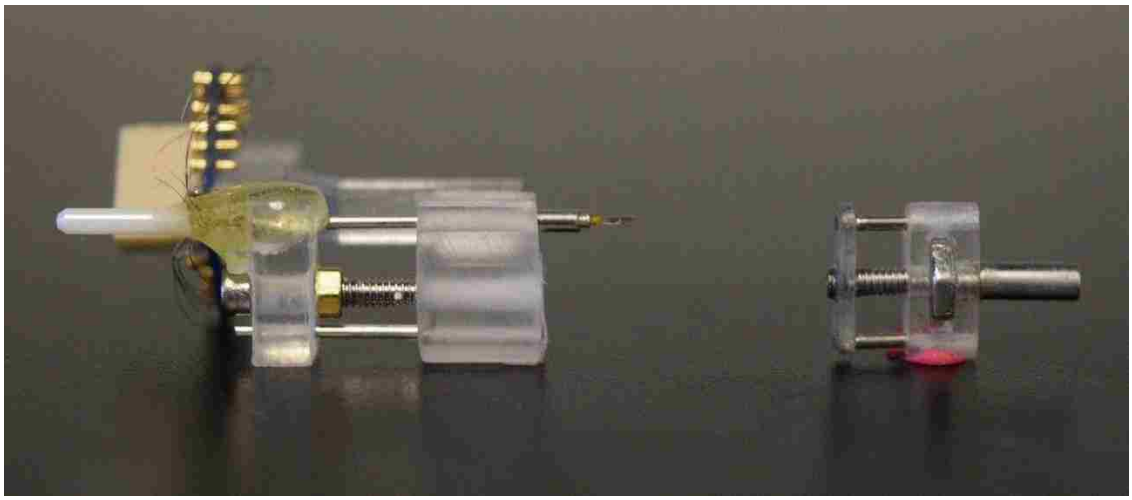

Figure 15: ***Comparison Between New and Previous Device*** *– Side-by-side comparison of the previous device (Left) to the new device (Right) showing significant reduction in size for the new device. Note: This is an image of the new device without the fibers implanted or EIB attached which makes it appear smaller than the final implanted device would be.*

Further validation testing would be required before implanting the new device including drive mechanism testing of fully constructed device, electrical conductance testing, and optical luminance testing. The drive mechanism must be tested to ensure that an appropriate amount of force is required to drive the fibers such that the nut will not spin on its own yet is easily driven by the researcher as to not apply excessive torsional force to the animal's skull. Since the new device was manufactured this month and the materials to fully construct the final device have not been delivered, these final validation tests will be conducted at a future date.

EXPERIMENTAL DESIGN AND RESULTS

BACKGROUND AND INTRODUCTION

The new device is intended for use in behavioral experiment in mice with the goal of improving the understanding of how brain regions communicate to drive behaviors associated with stress and addiction. Numerous studies have shown that persistent, inescapable stress increases the likelihood of drug abuse in humans (Piazza, 1990; Shaham, 1994; Kosten, 2000). The Chavkin Lab aims to investigate the neural mechanisms and pharmacology involved in this relationship between stress and addiction. In previous work, the Chavkin Lab has shown that the KOR/dynorphin system (described in Introduction) is responsible for the stress-induced increase in cocaine preference (Figure 16) (McLaughlin, 2003).

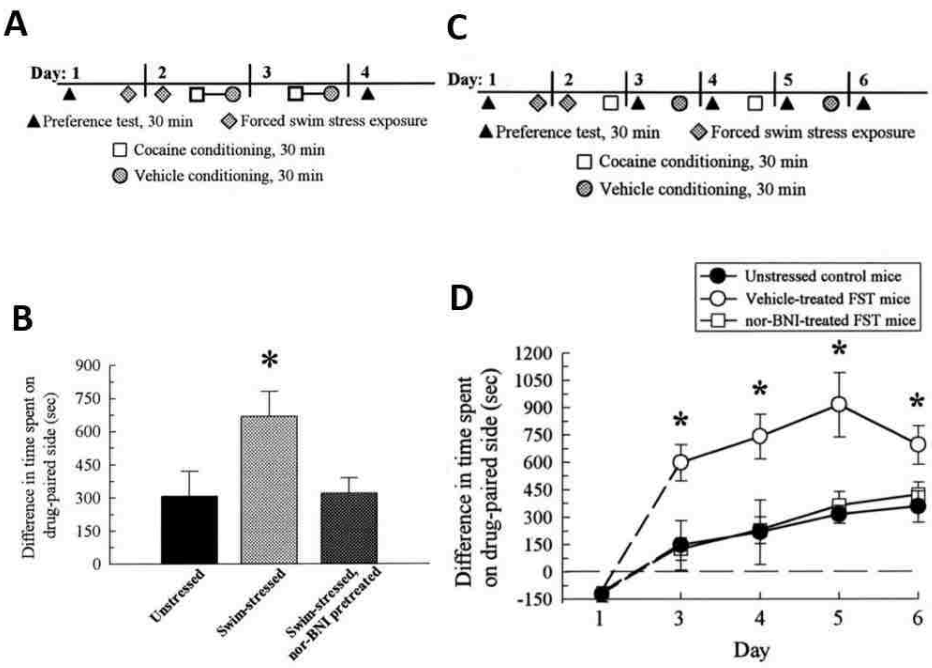

*Figure 16: **Stress-Induced Cocaine CPP is Blocked by nor-BNI** – nor-BNI, a KOR antagonist, blocks the stress-induced increase in cocaine preference in the conditioned place preference (CPP) task. A) Experimental paradigm for B. B) Forced swim stress (FSS) increases time spent on cocaine paired chamber during CPP, this effect is blocked by nor-BNI pretreatment. C) Experimental paradigm for D. D) The FSS induced reward potentiation is observed multiple days after initial stress exposure, the effect of nor-BNI continues throughout time course of experiment. (Adapted from McLaughlin et al., 2003).*

Further work demonstrated that activation of the KOR/dynorphin system is aversive (Land, 2008), and activation of KOR during cocaine conditioning results in an elimination of cocaine preference. Due to these results, it was theorized that activation of the KOR/dynorphin

system reduces the baseline hedonic state of the animal, allowing for a greater potential positive valence of the rewarding drug (Figure 17) (Bruchas, 2010).

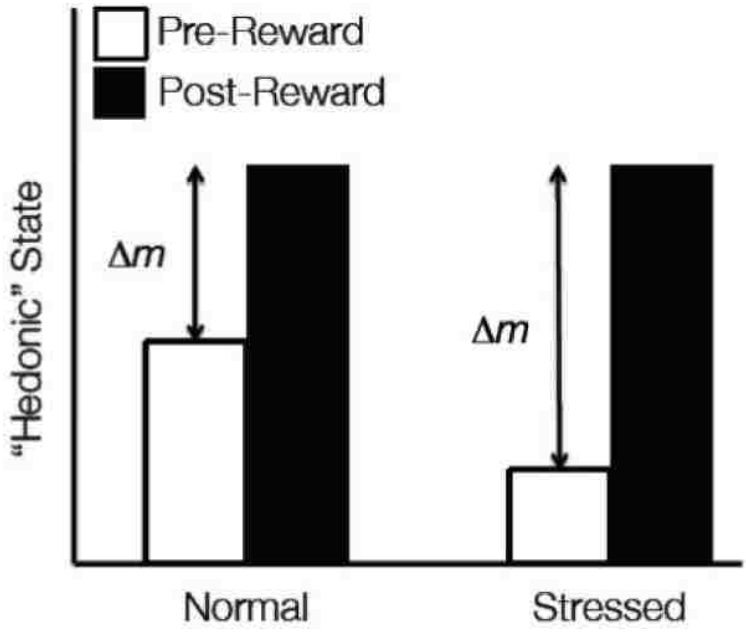

Figure 17: ***Change in Hedonic State Theory of KOR Mediated Reward*** *– Activation of the KOR/dynorphin system through stress increases dysphoria, negatively shifting the hedonic or 'mood' state of the animal. This downward shift in baseline creates larger potential positive valence for the drug of abuse, allowing the animal to experience more rewarding effects of the drug.*

In explanation of the 'reduced hedonic baseline' reward theory, the Chavkin Lab proposed a cellular mechanism. Since KOR is expressed on dopamine neurons and activation of the KOR/dynorphin system is aversive, the Chavkin Lab proposed that KOR keeps dopamine neurons in a prolonged inhibitory state which results in the reduction in hedonic state. In order to test this hypothesis, cellular level interrogation of the reward circuit during behavioral analysis is required. This interrogation involves real-time neural activity recording and modulation at multiple brain regions, necessitating a multi-site optrode. An overview of the designed experimental paradigm is described in brief below.

EXPERIMENTAL DESIGN

The basic experimental design proposed mimics the CPP paradigm from the predecessing experiments (outlined in Figure 16). The cocaine CPP paradigm is a four-day experiment (Figure 18). On the first day, an animal is placed into a 3-chamber apparatus and allowed to explore freely. The amount of time spent in each chamber is recorded and serves as the pre-test baseline.

After the pretest, the animal is exposed to a 15-minute forced swim stress (FSS). On the second day, the animal is exposed to a repeated forced swim stress (rFSS) paradigm in which it is forced to swim for 6 minutes, 4 times. After the rFSS the animal is conditioned by being injected with either cocaine (test group) or saline (control group) and placed into one chamber in the original 3-chamber apparatus, but is not allowed to explore the other chambers. The conditioning paradigm allows the animal to associate one of the chambers with the rewarding drug. On the third day, the animal is conditioned again with either cocaine or saline and placed in the same chamber as the previous day. On the final day, the animal is placed in the open apparatus and allowed to explore freely. The amount of time the animal spends in each chamber is recorded and compared to their pretest values to determine whether there was an increased the preference for the drug-paired chamber. A non-stressed control group is used in all experiments.

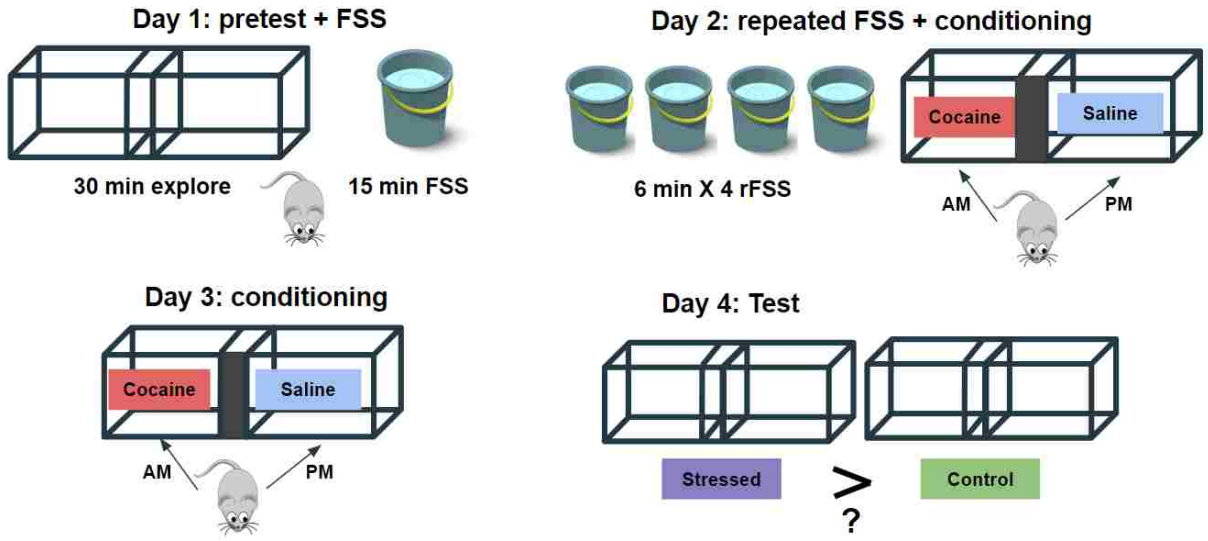

*Figure 18: **CPP Paradigm –** Schematic describing the 4-day cocaine conditioned place preference paradigm used in the behavioral experiments. Day 1: Pretest in 3-chamber apparatus followed by 15-minute forced swim stress. Day 2: Repeated forced swim stress followed by drug-paired chamber conditioning. Day 3: Repeated chamber conditioning. Day 4: Test day in open 3-chamber apparatus.*

Optogenetic Inhibition

This CPP paradigm is repeated with either an injection of U50-488, a selective KOR agonist or optogenetic inhibition of dopaminergic or serotonergic neurons in the VTA using Step-waveform inhibitory ChR (SwiChR) prior to conditioning. We would expect pretreatment with U50-488 to be aversive and cancel out the rewarding effects of cocaine, resulting in reduced or eliminated cocaine reward potentiation. To confirm the hypothesis that the KOR mediated

reward potentiation is due to a prolonged inhibition of dopaminergic neurons, we would expect the SwiChR inhibition to also result in a similar prevention of cocaine conditioning as seen with the U50-488 pretreated animals.

Electrophysiological Recording

To confirm that SwiChR is successfully inhibiting the dopamine neurons and to determine how that inhibition is affecting behavior, real-time electrophysiological recordings are required during the SwiChR inhibition. Recording the neural activity will allow us to determine whether the effects of SwiChR are due to a reduction in neural noise during the inhibition, or whether the effects are due to an increase in signal strength post-inhibition.

The use of electrophysiological recording during the behavioral experiments would also allow us to determine changed in the real-time neural activity during key behavioral time points. For example, we would be interested in investigating how the rate of neuronal firing changes when an animal enters a new chamber in the apparatus.

Multi-Site Recording

After determining the behavioral and cellular effect of optogenetic inhibition of dopamine neurons in the VTA prior to cocaine conditioning, we could use the dual-site modality of the device to investigate the circuit properties of the drug-induced behavioral changes. By exciting or inhibiting neurons in the VTA and recording simultaneously from the PFC or DRN, we could gain insights into the circuit level neural activity changes.

COMPUTATIONAL RESULTS

After collecting the electrophysiological neuronal activity data, we need a way to process and analyze the results. This requires the generation of a computational tool that can read the raw signal data, process the signal across a time scale, time-lock the signal to the recorded animal behavior, and perform post-processing analysis. Post-processing analysis entails determination of signal baseline, change from baseline calculations, and statistical analysis across trials. While the previously described data has not been collected using the new device, similar data could be used to begin developing the program that will be used to process the experimental data once it has been made available. Using the results from an adjacent experiment in the Chavkin Lab, a proof-of-concept computational program, with the aforementioned utilities was developed.

GCaMP Imaging

The related experiment in the Chavkin Lab uses GCaMP imaging to record *in vivo* neural activity signals. GCaMP is a calcium indicator protein that fluoresces when calcium concentrations increase. When a neuron is excited, the intracellular calcium concentration increases which causes GCaMP to fluoresce. The data collected from these experiments carries similar information to what will be recorded using the electrophysiology set-up; an increase in fluorescence represents an increase in neural activity using GCaMP just as an increase in voltage change represents an increase in neural activity using electrophysiology.

Using a fiber photometry set-up, the Chavkin Lab has recorded GCaMP signals from mice during behavioral experiments. During these experiments, the animal is able to move freely. Important behavioral time-points are recorded during the session to be time-locked with the neural activity during the signal processing step.

Computational Design

Before the GCaMP data could be processed in MATLAB, an intermediary program (TDT2TXT) was developed by Scott Ng-Evans to extract the data from the acquisition system format (.tdt) and export as a MATLAB readable format (.txt). Using the exported files, the GCaMP data was read into MATLAB (Appendix A). The data was down-sampled by a factor of 10 and sorted into two vectors (signal and time). Timestamps were then generated corresponding to the recorded animal behaviors.

The post-processing analysis began with calculating an activity baseline by taking the mean value of the data preceding the behavior of interest. The change from baseline was then calculated by taking the mean value of the data during the peak behavioral timepoint and subtracting the baseline value.

To evaluate the neural activity across behavioral trials, a 200 second window of data (50 seconds preceding behavioral timestamp, 150 second post behavior) was saved for each behavioral instance. The data were represented as a percent change by taking the absolute value of the data, subtracting the baseline value for each trial, and dividing by the baseline. A running average was used to smooth the data, then the mean and standard error were calculated across trials and plotted.

Results

Example traces of fluorescence data timestamped with important events from four separate behavioral trials can be seen in Figure 19. The recordings were from dopaminergic neurons in the VTA. In the top plot, there appears to be a gradual downwards drift of baseline after injection of U50-488 (black line) which we would expect if KOR activation causes a sustained inhibition of dopamine neurons. In all of the plots, the green line marks interaction with a food pellet. There appeared to be a stereotypical activity increase during each food pellet interaction. The effects of the other events (cocaine administration, interaction with nestlet, door opening, and administration of nor-BNI) are difficult to resolve through the noise of the recording.

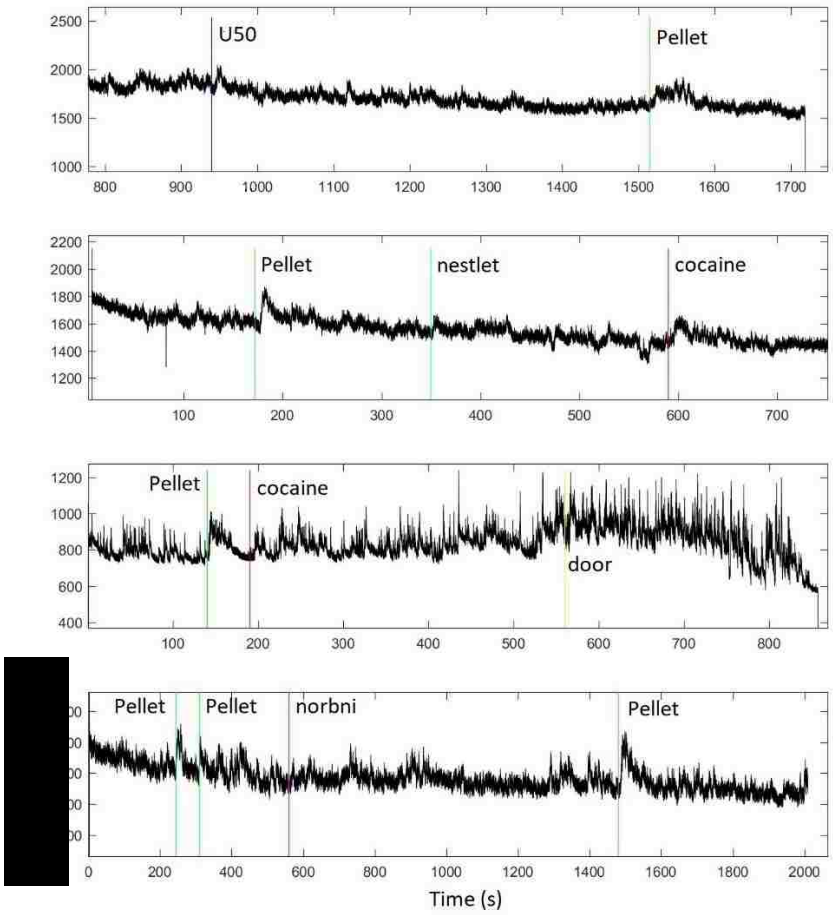

Figure 19: ***Event-Stamped GCaMP Fluorescence of Dopaminergic Neurons in VTA –*** *Raw data traces of GCaMP activity in dopaminergic neurons in the VTA during four recording sessions, timestamped with event tags. Top: Mouse 1 with injection of U50-488 and interaction with a food pellet. Second: Mouse 2, Day 1 with interactions with a food pellet, nestlet, and cocaine*

injection. Third: Mouse 2, Day 2 with interactions with a pellet, cocaine injection, and experimental disturbance of a door opening. Bottom: Mouse 3 with multiple food pellet interactions and an injection of nor-BNI.

Each instance of a pellet interaction from the four traces in Figure 19 were then averaged to determine a 'fingerprint' of neural activity during the behavior (Figure 20). The GCaMP activity showed a sharp increase within 5 seconds following the interaction with a gradual decline back to baseline over the course of 120 seconds. This activity was consistent across each interaction for every animal.

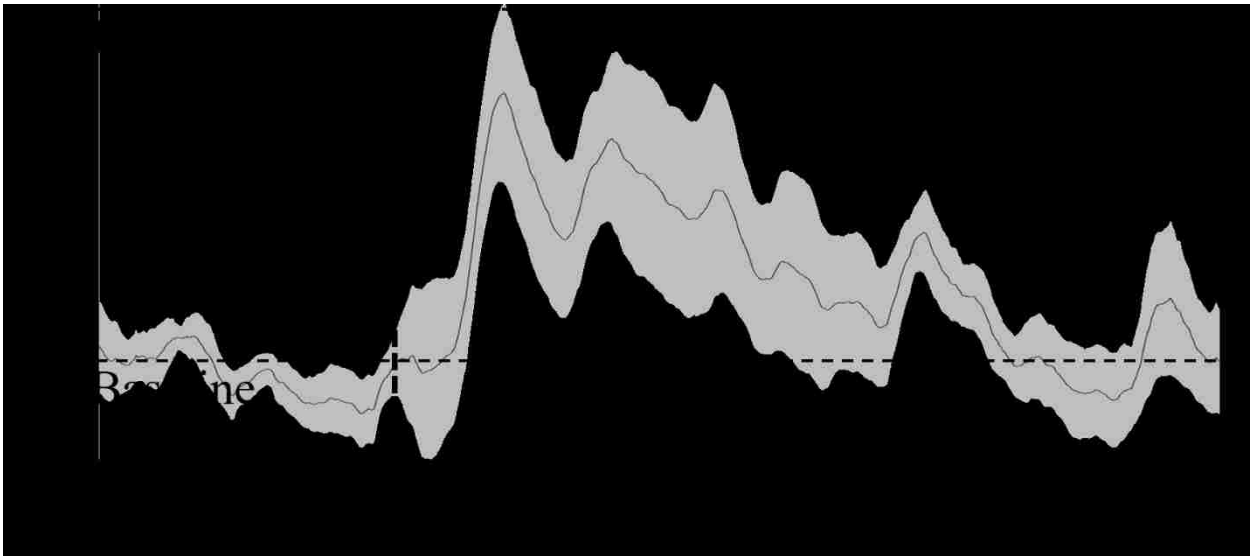

Figure 20: ***GCaMP Activity in Dopaminergic Neurons in VTA During Interaction with Food Pellet –*** *Plot showing the mean and standard error of GCaMP activity in dopaminergic neurons in the VTA of 3 separate mice across 6 interactions with a food pellet (1 from Mouse 1; 1 from Mouse 2, day 1; 1 from Mouse 2, day 2; 3 from Mouse 3). The GCaMP activity showed a sharp increase within 5 seconds following the interaction with a gradual decline back to baseline over the course of 120 seconds. This activity was consistent across each interaction for every animal.*

These results provide insight into the neuronal activity of dopaminergic neurons in the origin region of the mesolimbic pathway during a rewarding animal behavior. The mesolimbic pathway is responsible for processing reward signals and has been implicated as a major driver of drug abuse associate behaviors. While preliminary, these results offer a proof-of-concept validation that neural activity recordings can be time-locked to behavioral events and offer insight to the cellular activity underlying reward-driven behaviors.

CONCLUSION AND FUTURE WORK

AIMS ACHIEVED

The grounding purpose of this project was to develop the tools necessary to gain insights into the neural mechanisms and circuitry underlying the behaviors associated with drug abuse and addiction. There was a need for a multi-site *in vivo* recording device with the ability to modulate and record neuronal activity with high spatiotemporal resolution. In order to meet this need, significant design modification had to be made to reduce the weight and size of the current optrode device while maintaining high spatiotemporal resolution and fiber driving capabilities.

Optimization of Current Optrode Design

The first aim of this project was to optimize the current model device (Figure 10) to reduce the size and weight while maintaining the necessary functions for *in vivo* neural modulation and recording. Through multiple design iterations (Figures 11-13), a final design was manufactured (Figure 14), and validated against the predicate device for proof of optimization and maintenance of utility (Table 8). When comparing the new design to the previous design, all required device components were maintained while significant optimizations were made to the device size (Figure 15).

Development of Dual-Site Optrode

The second aim of this project was to develop a device capable of multi-site implantation while maintaining the necessary functions for *in vivo* neural modulation and recording. This aim was achieved concurrently to the first aim by taking the needs (Table 3), specifications (Table 4), and constraints (Table 5) for a dual-site device into consideration throughout the design process. The final design (Figures 13-14) successfully met the material, dimension, weight, and drive specification constraints for a dual-site optrode (Table 7).

Proof-Of-Concept Computational Design and Results

The final aim of this project was to develop the computational tools necessary to process and analyze neural activity data in the context of a behavioral experiment. While no data was generated from the device designed, data from a correlated experiment was used to develop a program capable of processing neural activity data, timestamping the data with behavioral

events, and analyzing the data for neural activity trends during behavior (Appendix A). The results from the processing and analysis of GCaMP activity in dopaminergic neurons in the VTA during interaction with a food pellet (Figures 19-20) offer a proof-of-concept validation that neural activity recordings can be time-locked to behavioral events and offer insight to the cellular activity underlying reward-driven behaviors.

UNFORESEEN BARRIERS

Throughout this process, a number of barriers delayed the progress of the project. While this was an independent research design project, collaboration with a number of other individuals was necessary to achieve the proposed aims. Through this process of collaboration, a number of roadblocks and project delays were met.

Machinist Delays

The primary delays were experienced while working with the micro-machinist. The machinist was based out of California, so all communication was done by email or phone and all materials had to be shipped by mail. Miscommunications over email resulted in design set-backs and unnecessary design iterations on both ends. Shipping delays also accounted for a large amount of the timeline set-backs.

The primary set-back was encountered when Prototype I (Figures 11-12) was deemed infeasible to manufacture and had to be redesigned. This redesign process took an additional 2.5 months during the 12-month timeline, significantly reducing timely progress. In the future, I will account for the likelihood of multiple design iteration phases when developing the project timeline.

Communication with Funding Personnel

Additional barriers were faced during communication with the funding and budget personnel for the project. Lack of timely information regarding available budget and appropriate spending strategies resulted in delays while ordering materials for the project. The primary delay occurred while ordering the required tooling. The budget personnel did not notify me of a vendor spending minimum until multiple weeks after the initial purchase request was sent. In the future, I will request a complete budget plan and guidelines for spending at the forefront of the project.

ACKNOWLEDGEMENTS

KEY PERSONNEL

I would like to thank Dr. Charles Chavkin for allowing me to work in his lab and for opening this field of research and inspiring the ground work and direction for this project. I would also like to thank my direct mentor, Dr. Antony Abraham, for his consistent devotion to ensuring the academic and personal success of his students and for his generosity with time and knowledge throughout this project. I also want to acknowledge key members of the Zweifel Lab: Larry Zweifel, Yong Jo, and Barbara Juarez, for their collaboration and sharing of knowledge, workspace, and materials. I also want to acknowledge my machinist, Kevin Willis, and his partner, Ken Brown, for their consultation during the device design process and generosity during prototyping. Finally, I want to thank my Bioengineering Co-Advisor, Dr. Alyssa Taylor, for her support of the project and ensurance that I was meeting all of the project requirements.

EQUIPMENT AND FACILITIES

The majority of this project was conducted in the University of Washington Health Sciences Building. The construction of the model devices were done in Larry Zweifel's lab using his equipment and materials. The equipment used included a dremel, soldering iron, dissecting microscope, and a number of hand tools. Some of the design process and prototyping was conducted in Makerspace at the University of Washington, for use of their 3D printers. The final device development was done in the Verus Design and Prototype workshop. Finally, Autodesk Inventor was used for the CAD modelling of the device and MATLAB was used for the computational development.

FUNDING

I would like to thank the National Institute of Mental Health, the National Institute on Drug Abuse, the UW Institute for Neural Engineering, the Washington Research Foundation, and the Levinson Emerging Scholar Award for funding the materials and spaces for this work. I would also like to thank the UW Computational Neuroscience Program for funding my research time and efforts. (R01 DA030074 (CC), T32 DA007278 (AA), and P50MH106428 (LZ, CC). The authors declare no financial conflict of interest.)

REFERENCES

Bruchas, M.R., Land, B.B. & Chavkin, C. (2010). The Dynorphin-Kappa Opioid System as a Modulator of Stress-induced and Pro-addictive Behaviors. *Brain Res.,* 1314C: 44.

Buchen, L. (2010). Neuroscience: Illuminating the brain. *Nature,* 465:26-28.

Center for Behavioral Health Statistics and Quality. (2016). Key substance use and mental health indicators in the United States: Results from the 2015 National Survey on Drug Use and Health (HHS Publication No. SMA 16-4984, NSDUH Series H-51). Retrieved from http://www.samhsa.gov/data/

Chavkin, C., & Ehrich, J. M. (2014). How does stress-induced activation of the kappa opioid system increase addiction risk? *Biological Psychiatry*, 76(10):760–762.

Goeders, N.E. (2002). Stress and cocaine addiction. *J Pharmacol Exp Ther.,* 301:785–789.

Gradinaru, V., Thompson, K.R., Zhang, F., Mogri, M., Kay, K., Schneider, M.B., & Deisseroth, K. (2007). Targeting and Read out Strategies for Fast Optical Neural Control In Vitro and In Vivo. *J. Neurosci.*, 27(52):14231–14238.

Institutional Animal Care and Use Committee Guidebook (2nd ed.). (2002) Bethesda, MD: Office of Laboratory Animal Welfare, NIH.

Kawakami, M. & Yamamura, K. (2008). Cranial bone morphometric study amound mouse strains. *BMC Evolutionary Biology,* 8:73

Kessler, R.C., Chiu, W.T., Demler, O., & Walters, E.E. (2005). Prevalence, severity, and comorbidity of twelve-month DSM-IV disorders in the National Comorbidity Survey Replication (NCSR). *Archives of General Psychiatry,* 62(6):617-27.

Kosten, T.A., Miserendino, M.J.D. & Kehoe, P. (2000) Enhanced acquisition of cocaine self-administration in adult rats with neonatal isolation stress experience. *Brain Res.,* 875:44–50.

Land, B.B., Bruchas, M.R., Lemos, J.C., Xu, M., Melief, E.J. & Chavkin, C. (2008). The dysphoric component of stress is encoded by activation of the dynorphin kappa-opioid system. *J Neurosci.,* 28(2):407-14.

Lowenstein, P.R. & Castro, M.G. (2001). Genetic engineering within the adult brain: implications for molecular approaches to behavioral neuroscience. *Physiol. Behav.,* 73(5):833-839.

McLaughlin, J.P., Marton-Popovici, M., & Chavkin, C. (2003) Kappa opioid receptor antagonism and prodynorphin gene disruption block stress-induced behavioral responses. *J Neurosci.,* 23:5674–5683.

Miyazaki, K.W., Miyazaki, K., Tanaka, K.F. Yamanaka, A., Takahashi, A., Tabuchi, S., & Doya, K. (2014). Optogenetic Activation of Dorsal Raphe Serotonin Neurons Enhances Patience for Future Rewards. *Current Biology*, 24(17):2033-2040.

Nicolelis, M.A. & Ribeiro, S. (2002). Multielectrode recording: the next steps. *Current Opinion in Neurobiology,* 12(5):602-606.

NIDA. (2015). Nationwide Trends. Retrieved from https://www.drugabuse.gov/publications/drugfacts/treatment-statistics.

Oztas, E. (2003) Neuronal tracing. *Neuroanatomy*, 2:2-5.

Paxinos, G. & Franklin, K. (2012). Paxinos and Franklin's the Mouse Brain in Stereotaxic Coordinates (4th ed.). Academic Press

Piazza, P.V., Deminiere, J.M., Le Moal, M. & Simon, H. (1990) Stress- and pharmacologically induced behavioral sensitization increases vulnerability to acquisition of amphetamine self-administration. *Brain Res.*, 514:22–26.

Rolls, A., Colas, D., Adamantidis, A., Carter, M., Lanre-Amos, T., Heller, H.G., & de Lecea, L. (2011). Optogenetic disruption of sleep continuity impairs memory consolidation. *PNAS,* 108(32):13305-13310.

Royer,, S., Zemelman, B.V., Barbic, M., Losonczy, A., Buzsáki, G., & Magee, J.C. (2010). Multi-array silicon probes with integrated optical fibers: light assisted perturbation and recording of local neural circuits in the behaving animal. *Eur J Neurosci*., 31(12): 2279–2291.

Shaham, Y. & Stewart, J. (1994). Exposure to mild stress enhances the reinforcing efficacy of intravenous heroin self-administration in rats. *Psychopharmacology (Berl),* 114:523–527.

Stefanik, M.T., Moussawi, K., Kupchik, Y.M., Smith, K.C., Miller, R.L., Huff, M.L., Deisseroth, K., Kalivas, P.W., & LaLumiere, R.T. (2012). Optogenetic inhibition of cocaine seeking in rats. *Addiction Biology*, 18(1):50-53.

Verus Design & Prototype. (2017). Consulting. Retrieved from http://www.v-dp.com.

Voigts, J., Siegle, J.H., Pritchett, D.L. & Moore, C.I. (2013). The flexDrive: an ultra-light implant for optical control and highly parallel chronic recording of neuronal ensembles in freely moving mice. *Front. Syst. Neurosci*., 7(8): 1662-5137.

APPENDICIES

APPENDIX A – MATLAB SCRIPT

Data Read, Sort, and Timestamp

```
%% Mouse 18 %%
%Pellet: 255 - 264
%Pellet approach: 271
%New Pellet: 310 - 330
%norbni: 560
%New Pellet: 1480 - 1550

filename_text_18 = 'Mouse-180418-112035_dvbg.txt';

data_text_18 = textread(filename_text_18,'%s');
data_text_18 = str2double(data_text_18);

[time_18,sig_18] = separate(data_text_18);

timelin_18 = linspace(0,data_text_18(end-1)*60,length(sig_18));

figure(1)
subplot(4,1,4)
plot(timelin_18,sig_18, 'k')
hold on
plot([245,245],[min(sig_18),max(sig_18)], 'g')
plot([310,310],[min(sig_18),max(sig_18)], 'g')
plot([1480,1480],[min(sig_18),max(sig_18)], 'g')
plot([560,560],[min(sig_18),max(sig_18)], 'm')
```

Down-Sample and Separate Signal from Time

```
function [time, sig] = separate(data)
    time = [];
    sig = [];
    for i = 1:20:length(data)-1
        time = [time,data(i)];
        sig = [sig,data(i+1)];
    end
end
```

Data Read, Sort, and Timestamp

```
preNorbni_18 = mean(sig_18(460*101:560*101));
```

```
postNorbni_18 = mean(sig_18(560*101:660*101)); %how long should
this last?
changeNorbni_18 = postNorbni_18 - preNorbni_18;

prePellet_18_1 = mean(sig_18(155*101:255*101));
postPellet_18_1 = mean(sig_18(255*101:305*101));
changePellet_18_1 = postPellet_18_1 - prePellet_18_1;

pelletSig_18_1 = (sig_18(205*101:355*101)-
prePellet_18_1)./prePellet_18_1;
pelletTime_18_1 = linspace(0,200,length(pelletSig_18_1));
plot(pelletTime_18_1, pelletSig_18_1)

prePellet_18_2 = mean(sig_18(264*101:310*101));
postPellet_18_2 = mean(sig_18(310*101:360*101));
changePellet_18_2 = postPellet_18_2 - prePellet_18_2;

pelletSig_18_2 = (sig_18(260*101:410*101)-
prePellet_18_2)./prePellet_18_2;
pelletTime_18_2 = linspace(0,200,length(pelletSig_18_2));
plot(pelletTime_18_2, pelletSig_18_2)

prePellet_18_3 = mean(sig_18(1380*101:1480*101));
postPellet_18_3 = mean(sig_18(1480*101:1530*101));
changePellet_18_3 = postPellet_18_3 - prePellet_18_3;

pelletSig_18_3 = (sig_18(1430*101:1580*101)-
prePellet_18_3)./prePellet_18_3;
pelletTime_18_3 = linspace(0,200,length(pelletSig_18_3));
plot(pelletTime_18_3, pelletSig_18_3)
```

Post-Processing

```
pelletAll = [pelletSig_16, pelletSig_17_1, pelletSig_17_2,...
    pelletSig_18_1, pelletSig_18_2, pelletSig_18_3];

pelletTime = pelletTime_16;

pelletAllRun = [];

avgWind = 500;
half = avgWind/2;

for j = 1:size(pelletAll,2)
    for i = 1:size(pelletAll,1)-(avgWind-1)
        wind = pelletAll(i:i+avgWind-1,j);
```

```
        avg = mean(wind);
        sd = std(wind);
        pelletAllRun(i,j) = avg;
    end
end

pelletAvgRun = [];
pelletStdRun = [];

for i = 1:size(pelletAllRun,1)
    pelletAvgRun(i) = mean(pelletAllRun(i,1:end));
    pelletStdRun(i) = std(pelletAllRun(i,1:end));
end

pelletErrRun = pelletStdRun./sqrt(6);

pelletTimeRun = pelletTime_16(1:length(pelletAvgRun));

figure(2)
subplot(3,1,3)
plot(pelletTimeRun, pelletAvgRun,'k-');
hold on
fill([pelletTimeRun flip(pelletTimeRun)],[pelletAvgRun-
pelletErrRun
flip(pelletAvgRun+pelletErrRun)],'k','LineStyle','none')
hold on
alpha(0.25)
hold off;

plot(pelletTimeRun, pelletAvgRun,'k-');
hold on
fill([pelletTimeRun flip(pelletTimeRun)],[pelletAvgRun-
pelletErrRun
flip(pelletAvgRun+pelletErrRun)],'k','LineStyle','none')
hold on
alpha(0.25)
hold off;
xlabel('Time (s)')
ylabel('DeltaF/F')
```

APPENDIX II – CODE DOCUMENTATION

DRL CODE

```
clear all; close all; clc;

%1-8 = dkt4
%9-16 = mc23

dateV = [204,205,208,209,210,211,215,216,217,218,...
    219,222,223,224,225,226,229,301,302,303,304,...
    307,308,309,310];

for d = 1:25

    date = dateV(d);

    activeNP = 0;
    reinNP = 0;
    headEnt = 0;
    resets = 0;
    irt = 0;

    for r = 1:2
        filename = strcat('0',num2str(date),'_',num2str(r),'.txt');
        data = textread(filename,'%s');

        IndexActNP = strfind(data, 'ActiveNosePokeTimes(sec)');
        IndexActNP = find(not(cellfun('isempty', IndexActNP)));

        IndexReinNP = strfind(data, 'ReinforcedTimes(sec)');
        IndexReinNP = find(not(cellfun('isempty', IndexReinNP)));

        IndexHeadEnt = strfind(data, 'HeadEntryTimes(sec)');
        IndexHeadEnt = find(not(cellfun('isempty', IndexHeadEnt)));

        IndexReset = strfind(data, 'DRLresetTimes(sec)');
        IndexReset = find(not(cellfun('isempty', IndexReset)));

        IndexIRT = strfind(data, 'IRTs(sec)');
        IndexIRT = find(not(cellfun('isempty', IndexIRT)));

        IndexEnd = strfind(data, 'Number_of_Bursts');
        IndexEnd = find(not(cellfun('isempty', IndexEnd)));

        %Active Nose Poke
        for i = 1:8
            start = IndexActNP(i)+1;
            last = IndexReinNP(i)-1;
            addCell = data(start:last);
            lengthAdd = size(addCell,1);
            lengthPrior = size(activeNP,1);
            add = zeros (lengthAdd,1);
            for j = 1 : lengthAdd
```

```
51.                 add(j) = str2double(addCell{j});
52.             end
53.             if lengthAdd > lengthPrior
54.                 padlength = lengthAdd - lengthPrior;
55.                 padwidth = size(activeNP,2);
56.                 padMatrix = zeros(padlength,padwidth);
57.                 activeNP = [activeNP;padMatrix];
58.             else
59.                 padlength = lengthPrior - lengthAdd;
60.                 padVector = zeros(padlength,1);
61.                 add = [add;padVector];
62.             end
63.             activeNP = [activeNP, add];
64.         end
65.
66.         %Reinforced Nose Poke
67.         for i = 1:8
68.             start = IndexReinNP(i)+1;
69.             last = IndexHeadEnt(i)-1;
70.             addCell = data(start:last);
71.             lengthAdd = size(addCell,1);
72.             lengthPrior = size(reinNP,1);
73.             add = zeros (lengthAdd,1);
74.             for j = 1 : lengthAdd
75.                 add(j) = str2double(addCell{j});
76.             end
77.             if lengthAdd > lengthPrior
78.                 padlength = lengthAdd - lengthPrior;
79.                 padwidth = size(reinNP,2);
80.                 padMatrix = zeros(padlength,padwidth);
81.                 reinNP = [reinNP;padMatrix];
82.             else
83.                 padlength = lengthPrior - lengthAdd;
84.                 padVector = zeros(padlength,1);
85.                 add = [add;padVector];
86.             end
87.             reinNP = [reinNP, add];
88.         end
89.
90.         %Head Entries
91.         for i = 1:8
92.             start = IndexHeadEnt(i)+1;
93.             last = IndexReset(i)-1;
94.             addCell = data(start:last);
95.             lengthAdd = size(addCell,1);
96.             lengthPrior = size(headEnt,1);
97.             add = zeros (lengthAdd,1);
98.             for j = 1 : lengthAdd
99.                 add(j) = str2double(addCell{j});
100.                end
101.                if lengthAdd > lengthPrior
102.                    padlength = lengthAdd - lengthPrior;
103.                    padwidth = size(headEnt,2);
```

```
104.                     padMatrix = zeros(padlength,padwidth);
105.                     headEnt = [headEnt;padMatrix];
106.                 else
107.                     padlength = lengthPrior - lengthAdd;
108.                     padVector = zeros(padlength,1);
109.                     add = [add;padVector];
110.                 end
111.                 headEnt = [headEnt, add];
112.             end
113.
114.             %Resets
115.             for i = 1:8
116.                 start = IndexReset(i)+1;
117.                 last = IndexIRT(i)-1;
118.                 addCell = data(start:last);
119.                 lengthAdd = size(addCell,1);
120.                 lengthPrior = size(resets,1);
121.                 add = zeros (lengthAdd,1);
122.                 for j = 1 : lengthAdd
123.                     add(j) = str2double(addCell{j});
124.                 end
125.                 if lengthAdd > lengthPrior
126.                     padlength = lengthAdd - lengthPrior;
127.                     padwidth = size(resets,2);
128.                     padMatrix = zeros(padlength,padwidth);
129.                     resets = [resets;padMatrix];
130.                 else
131.                     padlength = lengthPrior - lengthAdd;
132.                     padVector = zeros(padlength,1);
133.                     add = [add;padVector];
134.                 end
135.                 resets = [resets, add];
136.             end
137.
138.             %Inter response time
139.             for i = 1:8
140.                 start = IndexIRT(i)+1;
141.                 last = IndexEnd(i)-1;
142.                 addCell = data(start:last);
143.                 lengthAdd = size(addCell,1);
144.                 lengthPrior = size(irt,1);
145.                 add = zeros (lengthAdd,1);
146.                 for j = 1 : lengthAdd
147.                     add(j) = str2double(addCell{j});
148.                 end
149.                 if lengthAdd > lengthPrior
150.                     padlength = lengthAdd - lengthPrior;
151.                     padwidth = size(irt,2);
152.                     padMatrix = zeros(padlength,padwidth);
153.                     irt = [irt;padMatrix];
154.                 else
155.                     padlength = lengthPrior - lengthAdd;
156.                     padVector = zeros(padlength,1);
```

```
                    add = [add;padVector];
                end
                irt = [irt, add];
            end

        end

        eval(['activeNP_0' num2str(date) '=activeNP(1:end,2:end);'])
        eval(['reinNP_0' num2str(date) '=reinNP(1:end,2:end);'])
        eval(['headEnt_0' num2str(date) '=headEnt(1:end,2:end);'])
        eval(['resets_0' num2str(date) '=resets(1:end,2:end);'])
        eval(['IRT_0' num2str(date) '=irt(1:end,2:end);'])
    end

    percentError = zeros(length(dateV),i*r);
    numResets = zeros(length(dateV),i*r);
    numReins = zeros(length(dateV),i*r);
    numHeadEnts = zeros(length(dateV),i*r);

    for d = 1:length(dateV)
        date = dateV(d);

        eval(['reset = resets_0' num2str(date) ';'])
        eval(['rein = reinNP_0' num2str(date) ';'])
        eval(['headEnts = headEnt_0' num2str(date) ';']);

        for a = 1:i*r
            resetAnimal = reset(1:end,a);
            resetAnimal(resetAnimal==0) = [];
            numReset = length(resetAnimal);
            numResets(d,a) = numReset;

            reinAnimal = rein(1:end,a);
            reinAnimal(reinAnimal==0) = [];
            numRein = length(reinAnimal);
            numReins(d,a) = numRein;

            headEntAnimal = headEnts(1:end,a);
            headEntAnimal(headEntAnimal==0) = [];
            numHeadEnt = length(headEntAnimal);
            numHeadEnts(d,a) = numHeadEnt;

            percentError(d,a) = numReset/(numReset+numRein);
        end
    end

    totalNum = numResets + numReins;

    for d = 1:length(dateV)
        for z = 1:size(totalNum,2)
            t = totalNum(d,z);
            tempRes = strcat('resets_0',num2str(dateV(d)));
            eval(['res = ' tempRes '(1:end,z);'])
```

```
210.                res(res==0) = [];
211.                tempRei = strcat('reinNP_0',num2str(dateV(d)));
212.                eval(['rei = ' tempRei '(1:end,z);'])
213.                rei(rei==0) = [];
214.                tempBurst = strcat('IRT_0',num2str(dateV(d)));
215.                eval(['burst = ' tempBurst '(1:end,z);'])
216.
217.                for n = 1:length(burst)
218.                    if burst(n) == 0
219.                        burst(n) = 9;
220.                    else if burst(n) <= 1
221.                            burst(n) = 1;
222.                        else
223.                            burst(n) = 0;
224.                        end
225.                    end
226.                end
227.
228.                correct = zeros(1, length(res)+length(rei));
229.
230.                j = 0;
231.                c = 1;
232.                w = 1;
233.                n = 1;
234.                while j==0
235.                    if c <= length(rei)
236.                        if w <= length(res)
237.                            if rei(c) < res(w)
238.                                correct(n) = 1;
239.                                c = c+1;
240.                                n = n+1;
241.                            else
242.                                w = w+1;
243.                                n = n+1;
244.                            end
245.                        else
246.                            j = 1;
247.                            for m = n:t
248.                                correct(m) = 1;
249.                            end
250.                        end
251.                    else
252.                        j = 1;
253.                    end
254.                end
255.
256.                %1 = burst after correct
257.                %2 = burst after incorrect
258.                %3 = bursts after correct burst
259.                %4 = bursts after incorrect burst
260.
261.                diff = zeros(1,length(correct)-1);
262.                n = 1;
```

```
263.                while n <= length(diff)
264.                    if correct(n) == 1
265.                        if burst(n) == 1
266.                            diff(n) = 1;
267.                            for n = n+1:length(diff)
268.                                if correct(n) == 1
269.                                    break
270.                                else if burst(n) == 1
271.                                    diff(n) = 3;
272.                                    else
273.                                        break
274.                                    end
275.                                end
276.                            end
277.                        end
278.                        n = n+1;
279.
280.                    else if correct(n) == 0
281.                            if burst(n) == 1
282.                                diff(n) = 2;
283.                                for n = n+1:length(diff)
284.                                    if correct(n) == 1
285.                                        break
286.                                    else if burst(n) == 1
287.                                        diff(n) = 4;
288.                                        else
289.                                            break
290.                                        end
291.                                    end
292.                                end
293.                            end
294.                        end
295.                        n = n+1;
296.                    end
297.                end
298.                for row = 1:length(diff)
299.                    burstAnalysis(row,z) = diff(1,row);
300.                end
301.                for row = length(diff)+1:length(burst)
302.                    burstAnalysis(row,z) = 9;
303.                end
304.            end
305.            eval(['burstAnalysis_0' num2str(dateV(d)) '= burstAnalysis;'])
306.            clear burstAnalysis;
307.        end


488.        drl5animals = [1,2,3,4];
489.        dkt4animals = [5,6,7,8];
490.        mc23animals = [1,4,5,6,8];
491.
492.        ySal = 0;
493.        yTest = -.5;
494.        %DRL5 SALINE
```

```
495.        for a = 1:length(drl5animals)
496.            ySal = ySal + 2;
497.            xReinSal = DRL5reinNP_0324(1:end, drl5animals(a));
498.            xResetSal = DRL5resets_0324(1:end, drl5animals(a));
499.
500.            for i = 1:length(xReinSal)
501.                if xReinSal(i) == 0
502.                    xReinSal = xReinSal(1:i-1);
503.                    break
504.                end
505.            end
506.
507.            for i = 1:length(xResetSal)
508.                if xResetSal(i) == 0
509.                    xResetSal = xResetSal(1:i-1);
510.                    break
511.                end
512.            end
513.
514.            yReinSal = ones(length(xReinSal),1);
515.            yResetSal = ones(length(xResetSal),1);
516.
517.            for y = 1:length(xReinSal)
518.                yReinSal(y) = ySal;
519.            end
520.
521.            for y = 1:length(xResetSal)
522.                yResetSal(y) = ySal;
523.            end
524.
525.            plot(xReinSal,yReinSal, 'b.', 'MarkerSize', 12)
526.            hold('on');
527.            plot(xResetSal,yResetSal, 'b+', 'MarkerSize',12)
528.            hold('on');
529.        end
```

DELAYED ALTERNATION CODE

```
%Delayed Alternation Task
%Chavkin Lab 2017
%Author: Mackenzie Andrews

clear all; close all; clc;

load('C:\Users\Kenzie Marae\Desktop\Lab Materials\MATLAB\Delayed Alternation Data\loadin_workspaceData.mat');

%%
clear all; close all; clc;

%Enter File Path Containing Raw Data
folderPath = 'C:\Users\Kenzie Marae\Desktop\Lab Materials\MATLAB\Delayed Alternation Data\raw_data';
cd(folderPath);

files      = dir('**');
files(1:2) = [];
% files(n:end) = [];
numfiles   = numel(files);

[runs(1:numfiles).filename] = files.name;

for filenum = 1:numfiles
    filename = files(filenum).name;

    loadin = regexp( fileread(filename), '\n', 'split').';
    split1 = regexp(loadin,'\s+','split');

    rawData = cell(length(split1),8);
    for i = 1:length(split1)
        D = split1{i};
        for j = 1:length(D)
                insert = D{j};
                if isempty(insert) == 1
                    insert = NaN;
                elseif isempty(str2num(insert)) == 0
                    insert = str2num(insert);
                end
                rawData{i,j} = insert;
        end
    end
    runs(filenum).rawdata = rawData;

    runs(filenum).datalocations.rowB = find(strcmp(rawData(:,1),'B:'));
    runs(filenum).datalocations.rowC = find(strcmp(rawData(:,1),'C:'));
    runs(filenum).datalocations.rowE = find(strcmp(rawData(:,1),'E:'));
    runs(filenum).datalocations.rowT = find(strcmp(rawData(:,1),'T:'));

    dateTemp = split1{4}{3};
    date = strcat([dateTemp(7:8),dateTemp(1:2),dateTemp(4:5)]);
    runs(filenum).date = date;

    numBox = length(runs(filenum).datalocations.rowB);
    runs(filenum).numboxes = numBox;

```

```
    %Create B vector
    for boxnum = 1:numBox
        Braw =
rawData(runs(filenum).datalocations.rowB(boxnum)+1:runs(filenum).datalocations.rowC(box
num)-1,3:7);
        Btemp = [];
        for j = 1:size(Braw,1)
            current = Btemp;
            next5   = Braw(j,:);
            Btemp   = [current,next5];
        end
        runs(filenum).animal(boxnum).parseddata.B = Btemp;
    end

    %Create E vector
    for boxnum = 1:numBox
        Eraw =
rawData(runs(filenum).datalocations.rowE(boxnum)+1:runs(filenum).datalocations.rowT(box
num)-1,3:7);
        Etemp = [];
        for j = 1:size(Eraw,1)
            current = Etemp;
            next5   = Eraw(j,:);
            Etemp   = [current,next5];
        end
        runs(filenum).animal(boxnum).parseddata.E = Etemp;
    end

    %Create T vector
    for boxnum = 1:numBox
        Traw =
rawData(runs(filenum).datalocations.rowT(boxnum)+1:runs(filenum).datalocations.rowT(box
num)+ceil(length(runs(filenum).animal(boxnum).parseddata.E)/5),3:7);
        Ttemp = [];
        for j = 1:size(Traw,1)
            current = Ttemp;
            next5   = Traw(j,:);
            Ttemp   = [current,next5];
        end
        runs(filenum).animal(boxnum).parseddata.T = Ttemp;
    end
end

for f = 1:numfiles
    numBoxes = runs(f).numboxes;
    for ani = 1:numBoxes
        if isempty(runs(f).animal(ani).parseddata.B) == 1
            break;
        else
            Bcur = runs(f).animal(ani).parseddata.B;

            runs(f).animal(ani).totals.right_presses        = Bcur(1);
            runs(f).animal(ani).totals.left_presses         = Bcur(2);
            runs(f).animal(ani).totals.correct_alts         = Bcur(3);
            runs(f).animal(ani).totals.missed_alts          = Bcur(4);
            runs(f).animal(ani).totals.pellets_deliv        = Bcur(8);
            runs(f).animal(ani).totals.correct_alts_right   = Bcur(9);
            runs(f).animal(ani).totals.correct_alts_left    = Bcur(10);
            runs(f).animal(ani).totals.incorrect_alts_right = Bcur(11);
            runs(f).animal(ani).totals.incorrect_alts_left  = Bcur(12);
        end
```

```
111.    end
112.end
113.%%
114.flag = [];
115.for f = 1:numfiles
116.
117.        id = regexp(runs(f).filename(9:end),'\d+','match');
118.        if length(id) == 0;
119.            flag = [flag,f];
120.        else
121.        runs(f).animalID = str2num(id{1});
122.        end
123.
124.end
125.
126.for i = 1:length(flag)
127.    a{i,1} = {runs(flag(i)).filename};
128.end
129.
130.
131.%% PUT NUM CORRECT,MISSED; % CORRECT; TOTAL RESPONSES INTO STRUCTURE
132.dates = unique({runs.date});
133.
134.clear sortByDate
135.for i = 1:length(dates)
136.    sortByDate(i).dates = dates{i};
137.end
138.
139.numcorrect        = cell(length(dates),8,2);
140.nummissed         = cell(length(dates),8,2);
141.percentcorrect    = cell(length(dates),8,2);
142.totalresponses    = cell(length(dates),8,2);
143.rightpresses      = cell(length(dates),8,2);
144.leftpresses       = cell(length(dates),8,2);
145.correctaltsright  = cell(length(dates),8,2);
146.correctaltsleft   = cell(length(dates),8,2);
147.incorrectaltsright= cell(length(dates),8,2);
148.incorrectaltsleft = cell(length(dates),8,2);
149.
150.for f = 1:numfiles
151.    for d = 1:length(dates)
152.        if runs(f).date == dates{d}
153.            for id = 1:8
154.                if runs(f).animalID == id
155.                    for b = 1:runs(f).numboxes
156.
157.                        numcorrect{d,id,b}         =
    cell2mat(runs(f).animal(b).totals.correct_alts);
158.                        nummissed{d,id,b}          =
    cell2mat(runs(f).animal(b).totals.missed_alts);
159.                        percentcorrect{d,id,b}     =
    numcorrect{d,id,b}./(numcorrect{d,id,b}+nummissed{d,id,b});
160.                        totalresponses{d,id,b}     =
    numcorrect{d,id,b}+nummissed{d,id,b};
161.                        rightpresses{d,id,b}       =
    cell2mat(runs(f).animal(b).totals.right_presses);
162.                        leftpresses{d,id,b}        =
    cell2mat(runs(f).animal(b).totals.left_presses);
163.                        correctaltsright{d,id,b}   =
    cell2mat(runs(f).animal(b).totals.correct_alts_right);
```

```
164.                        correctaltsleft{d,id,b}    =
    cell2mat(runs(f).animal(b).totals.correct_alts_left);
165.                        incorrectaltsright{d,id,b} =
    cell2mat(runs(f).animal(b).totals.incorrect_alts_right);
166.                        incorrectaltsleft{d,id,b}  =
    cell2mat(runs(f).animal(b).totals.incorrect_alts_left);
167.
168.
169.                        sortByDate(d).ID(id).box(b).numcorrect =
    cell2mat(runs(f).animal(b).totals.correct_alts);
170.                        sortByDate(d).ID(id).box(b).nummissed  =
    cell2mat(runs(f).animal(b).totals.missed_alts);
171.                        sortByDate(d).ID(id).box(b).percentcorrect =
    numcorrect{d,id,b}./(numcorrect{d,id,b}+nummissed{d,id,b});
172.                        sortByDate(d).ID(id).box(b).totalresponses =
    numcorrect{d,id,b}+nummissed{d,id,b};
173.
174.                        sortByDate(d).ID(id).box(b).parsedData =
    runs(f).animal(b).parseddata
175.                        sortByDate(d).ID(id).box(b).totals = runs(f).animal(b).totals
176.                    end
177.                end
178.            end
179.        end
180.    end
181.end
182.
183.%percentcorrect = cell2mat(numcorrect)./(cell2mat(numcorrect)+cell2mat(nummissed));
184.
185.%%
186.days = [1,2,3,4,5,8,9,10,11,12,15,16,17,18,19,22,29,30,32,33,36,37,38,39,40,...
187.
    50,51,52,53,54,66,67,68,71,72,73,74,75,78,79,80,81,82,83,84,85,86,87,88,89,90,...
188.        92,93,94,95,96,99,100,101,102,103,106,107,108,109,110,113,115,116,...
189.        120,121,123,124,127,128,129,130,131,134,135,136,137,...
190.        157,158,159,160,161,162,163,164,165,166,167,168]; %starting 10/29 = 1
191.%% PLOT % CORRECT AND TOTAL ALTERNATIONS FOR SALINE AND ACSF ANIMALS ON TEST DAYS
192.%norbni
193.figure;
194.for id = 1:4
195.    for box = 1:2
196.        for i = 1:size(percentcorrect,1)
197.            emp(i) = ~isempty(percentcorrect{i,id,box});
198.        end
199.        daysTemp = days.*emp;
200.        daysTemp(daysTemp == 0) = [];
201.        subplot(2,1,1)
202.        title('norbni');
203.        ylabel('Percent Correct');
204.        xlabel('Days');
205.        plot(daysTemp,[percentcorrect{:,id,box}]); hold on;
206.        subplot(2,1,2)
207.        ylabel('Total # Alternations');
208.        xlabel('Days');
209.        plot(daysTemp,[totalresponses{:,id,box}]); hold on;
210.        stackablevPCnorb{(id*2-rem(box,2)),:} = {percentcorrect{:,id,box}};
211.        stackablevTRnorb{(id*2-rem(box,2)),:} = {totalresponses{:,id,box}};
212.    end
213.end
214.
215.
```

```
for i = 1:94
    vec = [];
    for j = 1:8
        if isempty(cell2mat([stackablevPCnorb{j}(i)]));
             vec(end+1) = NaN;
        else
        vec(end+1) = cell2mat([stackablevPCnorb{j}(i)]);
        end
    end
    avPCnorb(i) = nanmean(vec);
    sdPCnorb(i) = nanstd(vec);
    sePCnorb(i) = sdPCnorb(i)/sqrt(length(vec));
end

daysplot = days;
daysplot(isnan(avPCnorb)) = [];
avPCnorb(isnan(avPCnorb)) = [];
sePCnorb(isnan(sePCnorb)) = [];

figure;
subplot(2,1,1)
plot(daysplot,avPCnorb,'k.');
hold on
errorbar(daysplot,avPCnorb,sePCnorb)
% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
   flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
hold on
% alpha(0.25)
plot([52,52],[0,1],'r');hold on; %stress
plot([80,80],[0,1],'b');hold on; %drug
plot([94,94],[0,1],'b');hold on; %drug
title('norbni')
xlabel('Days')
ylabel('Percent Correct')


for i = 1:94
    vec = [];
    for j = 1:8
        if isempty(cell2mat([stackablevTRnorb{j}(i)]));
             vec(end+1) = NaN;
        else
        vec(end+1) = cell2mat([stackablevTRnorb{j}(i)]);
        end
    end
    avTRnorb(i) = nanmean(vec);
    sdTRnorb(i) = nanstd(vec);
    seTRnorb(i) = sdTRnorb(i)/sqrt(length(vec));
end

daysplot = days;
daysplot(isnan(avTRnorb)) = [];
avTRnorb(isnan(avTRnorb)) = [];
seTRnorb(isnan(seTRnorb)) = [];

subplot(2,1,2)
plot(daysplot,avTRnorb,'k.');
hold on
errorbar(daysplot,avTRnorb,seTRnorb)
% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
   flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
```

```
275.hold on
276.% alpha(0.25)
277.plot([52,52],[0,100],'r');hold on; %stress
278.plot([80,80],[0,100],'b');hold on; %drug
279.plot([94,94],[0,100],'b');hold on; %drug
280.title('norbni')
281.xlabel('Days')
282.ylabel('Total # Alternations')
283.
284.
285.
286.%ACSF
287.figure;
288.for id = 5:8
289.    for box = 1:2
290.        for i = 1:size(percentcorrect,1)
291.            emp(i) = ~isempty(percentcorrect{i,id,box});
292.        end
293.        daysTemp = days.*emp;
294.        daysTemp(daysTemp == 0) = [];
295.        subplot(2,1,1)
296.        title('ACSF');
297.        ylabel('Percent Correct');
298.        xlabel('Days');
299.        plot(daysTemp,[percentcorrect{:,id,box}]); hold on;
300.        subplot(2,1,2)
301.        ylabel('Total # Alternations');
302.        xlabel('Days');
303.        plot(daysTemp,[totalresponses{:,id,box}]); hold on;
304.        stackablevPCacsf{((id-4)*2-rem(box,2)),:} = {percentcorrect{:,id,box}};
305.        stackablevTRacsf{((id-4)*2-rem(box,2)),:} = {totalresponses{:,id,box}};
306.    end
307.end
308.
309.
310.for i = 1:94
311.    vec = [];
312.    for j = 1:8
313.        if isempty(cell2mat([stackablevPCacsf{j}(i)]));
314.             vec(end+1) = NaN;
315.        else
316.        vec(end+1) = cell2mat([stackablevPCacsf{j}(i)]);
317.        end
318.    end
319.    avPCacsf(i) = nanmean(vec);
320.    sdPCacsf(i) = nanstd(vec);
321.    sePCacsf(i) = sdPCacsf(i)/sqrt(length(vec));
322.end
323.
324.daysplot = days;
325.daysplot(isnan(avPCacsf)) = [];
326.avPCacsf(isnan(avPCacsf)) = [];
327.sePCacsf(isnan(sePCacsf)) = [];
328.
329.figure;
330.subplot(2,1,1)
331.plot(daysplot,avPCacsf,'k.');
332.hold on
333.errorbar(daysplot,avPCacsf,sePCacsf)
334.% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
   flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
```

```
hold on
% alpha(0.25)
plot([86,86],[0,1],'r');hold on; %stress
plot([94,94],[0,1],'b');hold on; %drug
plot([115,115],[0,1],'b');hold on; %drug
title('ACSF')
xlabel('Days')
ylabel('Percent Correct')


for i = 1:94
    vec = [];
    for j = 1:8
        if isempty(cell2mat([stackablevTRacsf{j}(i)]));
             vec(end+1) = NaN;
        else
        vec(end+1) = cell2mat([stackablevTRacsf{j}(i)]);
        end
    end
    avTRacsf(i) = nanmean(vec);
    sdTRacsf(i) = nanstd(vec);
    seTRacsf(i) = sdTRacsf(i)/sqrt(length(vec));
end

daysplot = days;
daysplot(isnan(avTRacsf)) = [];
avTRacsf(isnan(avTRacsf)) = [];
seTRacsf(isnan(seTRacsf)) = [];

subplot(2,1,2)
plot(daysplot,avTRacsf,'k.');
hold on
errorbar(daysplot,avTRacsf,seTRacsf)
% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
   flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
hold on
% alpha(0.25)
plot([86,86],[0,100],'r');hold on; %stress
plot([94,94],[0,100],'b');hold on; %drug
plot([115,115],[0,100],'b');hold on; %drug
title('ACSF')
xlabel('Days')
ylabel('Total # Alternations')


pc = zeros(94,8,2)
for i = 1:8
    a = cell2mat(percentcorrect(:,i,1));
    b = cell2mat(percentcorrect(:,i,2));

    pc(1:length(a),i,1) = a;
    pc(1:length(b),i,2) = b;
end

%% Morphine Days
avPCmorphine = [];
sdPCmorphine = [];
sePCmorphine = [];

```

```
avTRmorphine = [];
sdTRmorphine = [];
seTRmorphine = [];


figure;
for id = 5:8
    for box = 1:2
        for i = 1:size(percentcorrect,1)
            emp(i) = ~isempty(percentcorrect{i,id,box});
        end
        daysTemp = days.*emp;
        daysTemp(daysTemp == 0) = [];
        subplot(2,1,1)
        title('ACSF');
        ylabel('Percent Correct');
        xlabel('Days');
        plot(daysTemp,[percentcorrect{:,id,box}]); hold on;
        subplot(2,1,2)
        ylabel('Total # Alternations');
        xlabel('Days');
        plot(daysTemp,[totalresponses{:,id,box}]); hold on;
        stackablevPCacsf{((id-4)*2-rem(box,2)),:} = {percentcorrect{:,id,box}};
        stackablevTRacsf{((id-4)*2-rem(box,2)),:} = {totalresponses{:,id,box}};
    end
end


for i = 83:94
    vec = [];
    for j = 1:8
        if isempty(cell2mat([stackablevPCacsf{j}(i)]));
             vec(end+1) = NaN;
        else
        vec(end+1) = cell2mat([stackablevPCacsf{j}(i)]);
        end
    end
    avPCmorphine(i) = nanmean(vec);
    sdPCmorphine(i) = nanstd(vec);
    sePCmorphine(i) = sdPCmorphine(i)/sqrt(length(vec));
end

daysplot = days;
daysplot(isnan(avPCmorphine)) = [];
daysplot(avPCmorphine == 0) = [];
avPCmorphine(isnan(avPCmorphine)) = [];
avPCmorphine(avPCmorphine == 0) = [];
sePCmorphine(isnan(sePCmorphine)) = [];
sePCmorphine(sePCmorphine == 0) = [];


figure;
subplot(2,1,1)
plot(daysplot,avPCmorphine,'k.');
hold on
errorbar(daysplot,avPCmorphine,sePCmorphine)
% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
   flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
hold on
% alpha(0.25)
```

```
455.% plot([86,86],[0,1],'r');hold on; %stress
456.% plot([94,94],[0,1],'b');hold on; %drug
457.% plot([115,115],[0,1],'b');hold on; %drug
458.title('Morphine Treatment')
459.xlabel('Days')
460.ylabel('Percent Correct')
461.
462.
463.
464.
465.for i = 83:94
466.    vec = [];
467.    for j = 1:8
468.        if isempty(cell2mat([stackablevTRacsf{j}(i)]));
469.             vec(end+1) = NaN;
470.        else
471.        vec(end+1) = cell2mat([stackablevTRacsf{j}(i)]);
472.        end
473.    end
474.    avTRmorphine(i) = nanmean(vec);
475.    sdTRmorphine(i) = nanstd(vec);
476.    seTRmorphine(i) = sdTRmorphine(i)/sqrt(length(vec));
477.end
478.
479.daysplot = days;
480.daysplot(isnan(avTRmorphine)) = [];
481.daysplot(avTRmorphine == 0) = [];
482.avTRmorphine(isnan(avTRmorphine)) = [];
483.avTRmorphine(avTRmorphine == 0) = [];
484.seTRmorphine(isnan(seTRmorphine)) = [];
485.seTRmorphine(seTRmorphine == 0) = [];
486.
487.
488.subplot(2,1,2)
489.plot(daysplot,avTRmorphine,'k.');
490.hold on
491.errorbar(daysplot,avTRmorphine,seTRmorphine)
492.% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
    flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
493.hold on
494.% alpha(0.25)
495.% plot([86,86],[0,100],'r');hold on; %stress
496.% plot([94,94],[0,100],'b');hold on; %drug
497.% plot([115,115],[0,100],'b');hold on; %drug
498.title('Morphine Treatment')
499.xlabel('Days')
500.ylabel('Total # Alternations')
501.%% PULL EVENT & TIME STAMP INFORMATION
502.
503.for f = 1:length(sortByDate)
504.    for id = 1:length(sortByDate(f).ID)
505.        if length(sortByDate(f).ID(id).box)>=1
506.            for b = 1:length(sortByDate(f).ID(id).box)
507.
508.                Emat = cell2mat(sortByDate(f).ID(id).box(b).parsedData.E);
509.                Tmat = cell2mat(sortByDate(f).ID(id).box(b).parsedData.T);
510.
511.                Emat(isnan(Emat)) = [];
512.                Tmat(isnan(Tmat)) = [];
513.
514.                right_lever_press_E = (Emat == 1);
```

```
515.                left_lever_press_E = (Emat == 2);
516.                correct_alts_right_E = (Emat == 15);
517.                correct_alts_left_E = (Emat == 16);
518.                incorrect_alts_right_E = (Emat == 17);
519.                incorrect_alts_left_E = (Emat == 18);
520.                delay_starts_E = (Emat == 23);
521.                delay_ends_E = (Emat == 24);
522.
523.                press_latency_vector{f,ani} = 10.*delay_ends_E + ...
524.                    2.*correct_alts_right_E + 2.*correct_alts_left_E + ...
525.                    1.*incorrect_alts_right_E + 1.*incorrect_alts_left_E;
526.
527.                sortByDate(f).ID(id).box(b).latencies.press_latency_vector = ...
528.                    10.*delay_ends_E + ...
529.                    2.*correct_alts_right_E + 2.*correct_alts_left_E + ...
530.                    1.*incorrect_alts_right_E + 1.*incorrect_alts_left_E;
531.
532.
533.                press_latency_binary_E = delay_ends_E + ...
534.                    correct_alts_right_E + correct_alts_left_E + ...
535.                    incorrect_alts_right_E + incorrect_alts_left_E;
536.
537.                right_lever_press_T = Tmat.*right_lever_press_E;
538.                left_lever_press_T = Tmat.*left_lever_press_E;
539.                correct_alts_right_T = Tmat.*correct_alts_right_E;
540.                correct_alts_left_T = Tmat.*correct_alts_left_E;
541.                incorrect_alts_right_T = Tmat.*incorrect_alts_right_E;
542.                incorrect_alts_left_T = Tmat.*incorrect_alts_left_E;
543.                delay_starts_T = Tmat.*delay_starts_E;
544.                delay_ends_T = Tmat.*delay_ends_E;
545.                press_latency_binary_T = Tmat.*press_latency_binary_E;
546.
547.                right_lever_press_T(right_lever_press_T == 0 |
    isnan(right_lever_press_T)) = [];
548.                left_lever_press_T(left_lever_press_T == 0 |
    isnan(left_lever_press_T)) = [];
549.                correct_alts_right_T(correct_alts_right_T == 0 |
    isnan(correct_alts_right_T)) = [];
550.                correct_alts_left_T(correct_alts_left_T == 0 |
    isnan(correct_alts_left_T)) = [];
551.                incorrect_alts_right_T(incorrect_alts_right_T == 0 |
    isnan(incorrect_alts_right_T)) = [];
552.                incorrect_alts_left_T(incorrect_alts_left_T == 0 |
    isnan(incorrect_alts_left_T)) = [];
553.                delay_starts_T(delay_starts_T == 0 | isnan(delay_starts_T)) = [];
554.                delay_ends_T(delay_ends_T == 0 | isnan(delay_ends_T)) = [];
555.                %press_latency_binary_T(press_latency_binary_T == 0 |
    isnan(press_latency_binary_T)) = [];
556.
557.                sortByDate(f).ID(id).box(b).latencies.rlp = right_lever_press_T;
558.                sortByDate(f).ID(id).box(b).latencies.llp = left_lever_press_T;
559.                sortByDate(f).ID(id).box(b).latencies.car = correct_alts_right_T;
560.                sortByDate(f).ID(id).box(b).latencies.cal = correct_alts_left_T;
561.                sortByDate(f).ID(id).box(b).latencies.iar = incorrect_alts_right_T;
562.                sortByDate(f).ID(id).box(b).latencies.ial = incorrect_alts_left_T;
563.                sortByDate(f).ID(id).box(b).latencies.dst = delay_starts_T;
564.                sortByDate(f).ID(id).box(b).latencies.den = delay_ends_T;
565.                sortByDate(f).ID(id).box(b).latencies.plb = press_latency_binary_T;
566.            end
567.        end
568.    end
```

```
end

%% MAKE RASTER PLOTS

for f = 70:length(sortByDate)
    for id = 1:length(sortByDate(f).ID)
        if length(sortByDate(f).ID(id).box)>=1
            for b = 1:length(sortByDate(f).ID(id).box)

                car = (sortByDate(f).ID(id).box(b).latencies.car)./(100*60);
                cal = (sortByDate(f).ID(id).box(b).latencies.cal)./(100*60);
                iar = (sortByDate(f).ID(id).box(b).latencies.iar)./(100*60);
                ial = (sortByDate(f).ID(id).box(b).latencies.ial)./(100*60);
                dst = (sortByDate(f).ID(id).box(b).latencies.dst)./(100*60);

                cr = plot([car;car],[ones(1,length(car)).*0;ones(1,length(car))],'b-
    ','linewidth',2); hold on;
                cl =
    plot([cal;cal],[ones(1,length(cal)).*0;ones(1,length(cal))],'Color',[1,.6,.2],'linewidt
    h',2); hold on;
                ir = plot([iar;iar],[ones(1,length(iar)).*0;ones(1,length(iar)).*-
    1],'b-','linewidth',2); hold on;
                il = plot([ial;ial],[ones(1,length(ial)).*0;ones(1,length(ial)).*-
    1],'Color',[1,.6,.2],'linewidth',2); hold on;
                fp = plot([dst;dst],[ones(1,length(dst)).*-
    0.2;ones(1,length(dst)).*0.2],'k-','linewidth',2); hold on;
                ce = plot([0,60],[0,0],'Color',[0,0,0,0.4])
                %                 fp =
    plot([dst],[zeros(1,length(dst))],'k.','MarkerSize',10); hold on;

                if length(cr) > 0 & length(cl) > 0
                    subset = [cr(1),cl(1)];
                    legend(subset,'Right Lever','Left Lever','Location','eastoutside')
                    legend boxoff
                elseif length(ir) > 0 & length(il) > 0
                    subset = [ir(1),il(1)];
                    legend(subset,'Right Lever','Left Lever','Location','eastoutside')
                    legend boxoff
                elseif length(cr) > 0 & length(il) > 0
                    subset = [cr(1),il(1)];
                    legend(subset,'Right Lever','Left Lever','Location','eastoutside')
                    legend boxoff
                elseif length(ir) > 0 & length(cl) > 0
                    subset = [ir(1),cl(1)];
                    legend(subset,'Right Lever','Left Lever','Location','eastoutside')
                    legend boxoff
                end


                names = {'Inorrect {\bf\downarrow}';'Correct {\bf\uparrow}'};
                xlabel('Time (min)');
                ylim([-1.2,1.2])
                ax = gca
                set(gca,'TickLength',[0,0])
                set(gca,'ytick',[-.5,.5],'yticklabel',names)
                ax.FontSize = 20;

                fig = gcf;
                set(gcf,'color','w');
                fig.PaperUnits = 'inches';
                fig.PaperPosition = [0 0 30 3];
```

```
623.                box off
624.%
625.                savename = strcat('C:\Users\Kenzie Marae\Desktop\Lab
    Materials\MATLAB\Delayed Alternation
    Data\rasterPlots2\',sortByDate(f).dates(1:end),'_',num2str(id),'_',num2str(b),'.jpg');
626.                saveas(gcf,savename)
627.                close
628.            end
629.        end
630.    end
631.end
632.
633.%% BIN LATENCIES, % CORRECT, and TOTAL RESPONSE
634.
635.numberOfBins = 4;
636.bins = [0:60/numberOfBins:60];
637.
638.% avg_correct_latency = zeros(numfiles,2);
639.% avg_incorrect_latency = zeros(numfiles,2);
640.
641.for f = 1:length(sortByDate)
642.    for id = 1:length(sortByDate(f).ID)
643.        if length(sortByDate(f).ID(id).box)>=1
644.            for b = 1:length(sortByDate(f).ID(id).box)
645.
646.
    sortByDate(f).ID(id).box(b).latencies.analysis.incorrect_lat_eventtimes = [];
647.                sortByDate(f).ID(id).box(b).latencies.analysis.correct_lat_eventtimes
    = [];
648.                sortByDate(f).ID(id).box(b).latencies.analysis.incorrect_lat = [];
649.                sortByDate(f).ID(id).box(b).latencies.analysis.correct_lat = [];
650.                for i = 1:numberOfBins
651.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(i).incorrect_lat = [];
652.                    sortByDate(f).ID(id).box(b).latencies.analysis.bins(i).correct_lat
    = [];
653.                end
654.
655.                press_latency =
    sortByDate(f).ID(id).box(b).latencies.press_latency_vector;
656.                latency_times = sortByDate(f).ID(id).box(b).latencies.plb;
657.
658.                for i = 1:length(press_latency)
659.                    incorrect_lat_eventtimes = [];
660.                    correct_lat_eventtimes = [];
661.                    incorrect_lat = [];
662.                    correct_lat = [];
663.
664.                    if press_latency(i) == 10
665.                        last_delay = latency_times(i);
666.                    end
667.                    if press_latency(i) == 1
668.
    sortByDate(f).ID(id).box(b).latencies.analysis.incorrect_lat(end+1) =...
669.                            latency_times(i)-last_delay;
670.
    sortByDate(f).ID(id).box(b).latencies.analysis.incorrect_lat_eventtimes(end+1) =...
671.                            last_delay;
672.
673.                        for j = 1:numberOfBins
```

```
674.                             if last_delay > bins(j)*60*100 & last_delay <=
    bins(j+1)*60*100
675.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_lat(end+1) =...
676.                                     latency_times(i)-last_delay;
677.                             end
678.                         end
679.                     end
680.                     if press_latency(i) == 2
681.
    sortByDate(f).ID(id).box(b).latencies.analysis.correct_lat(end+1) =...
682.                             latency_times(i)-last_delay;
683.
    sortByDate(f).ID(id).box(b).latencies.analysis.correct_lat_eventtimes(end+1) =...
684.                             last_delay;
685.                         for j = 1:numberOfBins
686.                             if last_delay > bins(j)*60*100 & last_delay <=
    bins(j+1)*60*100
687.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_lat(end+1) =...
688.                                     latency_times(i)-last_delay;
689.                             end
690.                         end
691.                     end
692.                 end
693.
694.                 sortByDate(f).ID(id).box(b).latencies.analysis.avg_correct_latency
    =...
695.                     mean(sortByDate(f).ID(id).box(b).latencies.analysis.correct_lat);
696.                 sortByDate(f).ID(id).box(b).latencies.analysis.avg_incorrect_latency
    =...
697.
    mean(sortByDate(f).ID(id).box(b).latencies.analysis.incorrect_lat);
698.
699.                 for j = 1:numberOfBins
700.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_latency_mean =...
701.
    mean(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_lat);
702.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_latency_mean =...
703.
    mean(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_lat);
704.
705.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_latency_std =...
706.
    std(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_lat);
707.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_latency_std =...
708.
    std(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_lat);
709.
710.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_latency_num =...
711.
    length(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_lat);
712.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_latency_num =...
713.
    length(sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_lat);
```

```
714.
715.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).totalNumResponses =...
716.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_latency_num +...
717.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).incorrect_latency_num;
718.
719.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).percentCorrect =...
720.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).correct_latency_num / ...
721.
    sortByDate(f).ID(id).box(b).latencies.analysis.bins(j).totalNumResponses;
722.                 end
723.
724.             end
725.         end
726.     end
727.end
728.
729.avg_cor_lat = zeros(5,12);
730.avg_inc_lat = zeros(5,12);
731.for f = 83:94
732.     for id = 6:8
733.         for b = 1:length(sortByDate(f).ID(id).box)
734.
735.           avg_cor_lat(((id-5)*2-2+b),f-82) =
    sortByDate(f).ID(id).box(b).latencies.analysis.avg_correct_latency;
736.           avg_inc_lat(((id-5)*2-2+b),f-82) =
    sortByDate(f).ID(id).box(b).latencies.analysis.avg_incorrect_latency;
737.         end
738.     end
739.end
740.
741.aniAvg_cor_lat = mean(avg_cor_lat);
742.aniSD_cor_lat = std(avg_cor_lat);
743.aniSE_cor_lat = aniSD_cor_lat./sqrt(size(avg_cor_lat,1));
744.
745.aniAvg_inc_lat = mean(avg_inc_lat);
746.aniSD_inc_lat = std(avg_inc_lat);
747.aniSE_inc_lat = aniSD_inc_lat./sqrt(size(avg_inc_lat,1));
748.
749.figure;
750.plot(daysplot,aniAvg_cor_lat,'g.');
751.hold on
752.errorbar(daysplot,aniAvg_cor_lat,aniSE_cor_lat)
753.% fill([daysplot flip(daysplot)],[avPCnorb-sePCnorb
    flip(avPCnorb+sePCnorb)],'k','LineStyle','none')
754.hold on
755.plot(daysplot,aniAvg_inc_lat,'r.');
756.hold on
757.errorbar(daysplot,aniAvg_inc_lat,aniSE_inc_lat)
758.% alpha(0.25)
759.% plot([86,86],[0,100],'r');hold on; %stress
760.% plot([94,94],[0,100],'b');hold on; %drug
761.% plot([115,115],[0,100],'b');hold on; %drug
762.title('Morphine Treatment Latencies')
763.xlabel('Days')
764.ylabel('Average Latencies')
765.
```

```
766.%%
767.norBNI = [1,2,3,4,5,6,7,8,27,28,29,30,34,35,36,37,41,42,43,44];
768.ACSF   = [9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,...
769.          31,32,33,38,39,40,45,46];
770.
771.norBNI_pre = [1,2,3,4];
772.norBNI_fss = [5,6,7,8];
773.
774.ACSF_pre = [12,13,14];
775.ACSF_fss = [15,16,17];
776.
777.norBNI_sal_10_1 = [27,28,29,30];
778.norBNI_sal_10_2 = [34,35,36,37];
779.norBNI_u50_10 = [41,42,43,44];
780.
781.ACSF_sal_10_1 = [31,32,33];
782.ACSF_sal_10_2 = [38,39,40];
783.ACSF_u50_10 = [45,46,47];
784.
785.stress_groups = {norBNI_pre, norBNI_fss; ACSF_pre, ACSF_fss};
786.drug_groups = {norBNI_sal_10_1, norBNI_sal_10_2, norBNI_u50_10;...
787.               ACSF_sal_10_1, ACSF_sal_10_2, ACSF_u50_10};
788.
789.%% DATES and GROUPS
790.acsf_pre = 47;
791.acsf_fss = 48;
792.
793.norB_pre = 28;
794.norB_fss = 29;
795.
796.norB_sal = 41;
797.norB_u50 = 42;
798.
799.all_sal  = 54;
800.all_u50  = 55;
801.
802.%%
803.%binPlot(1) = acsf_pre
804.%binPlot(2) = acsf_fss
805.%binPlot(3) = norB_pre
806.%binPlot(4) = norB_fss
807.%binPlot(5) = norB_sal
808.%binPlot(6) = norB_u50
809.%binPlot(7) = norB_sal (all)
810.%binPlot(8) = norB_u50 (all)
811.%binPlot(9) = acsf_sal (all)
812.%binPlot(10) = acsf_u50 (all)
813.
814.%data(1) = percent correct
815.%data(2) = number of responses
816.%data(3) = correct latency
817.%data(4) = incorrect latency
818.
819.for i = 1:numberOfBins
820.    for j = 1:4
821.        for k = 1:10
822.            binPlot(k).bin(i).data(j).vectors = [];
823.        end
824.    end
825.end
826.
```

```
827.day = acsf_pre;
828.plotnum = 1;
829.for i = 6:8
830.    for b = 1:length(sortByDate(day).ID(i).box)
831.        for Bin = 1:numberOfBins
832.            binPlot(plotnum).bin(Bin).data(1).vectors(end+1) =...
833.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
834.            binPlot(plotnum).bin(Bin).data(2).vectors(end+1) =...
835.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
836.            binPlot(plotnum).bin(Bin).data(3).vectors =...
837.                [binPlot(plotnum).bin(Bin).data(3).vectors, ...
838.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
839.            binPlot(plotnum).bin(Bin).data(4).vectors =...
840.                [binPlot(plotnum).bin(Bin).data(4).vectors, ...
841.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
842.        end
843.    end
844.
845.    for b = 1:length(sortByDate(day+1).ID(i).box)
846.        for Bin = 1:numberOfBins
847.            binPlot(plotnum+1).bin(Bin).data(1).vectors(end+1) =...
848.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
849.            binPlot(plotnum+1).bin(Bin).data(2).vectors(end+1) =...
850.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
851.            binPlot(plotnum+1).bin(Bin).data(3).vectors =...
852.                [binPlot(plotnum+1).bin(Bin).data(3).vectors, ...
853.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
854.            binPlot(plotnum+1).bin(Bin).data(4).vectors =...
855.                [binPlot(plotnum+1).bin(Bin).data(4).vectors, ...
856.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
857.        end
858.    end
859.end
860.
861.day = norB_pre;
862.plotnum = plotnum+2;
863.for i = 1:4
864.    for b = 1:length(sortByDate(day).ID(i).box)
865.        for Bin = 1:numberOfBins
866.            binPlot(plotnum).bin(Bin).data(1).vectors(end+1) =...
867.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
868.            binPlot(plotnum).bin(Bin).data(2).vectors(end+1) =...
869.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
870.            binPlot(plotnum).bin(Bin).data(3).vectors =...
871.                [binPlot(plotnum).bin(Bin).data(3).vectors, ...
872.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
873.            binPlot(plotnum).bin(Bin).data(4).vectors =...
874.                [binPlot(plotnum).bin(Bin).data(4).vectors, ...
875.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
```

```
876.        end
877.    end
878.
879.    for b = 1:length(sortByDate(day+1).ID(i).box)
880.        for Bin = 1:numberOfBins
881.            binPlot(plotnum+1).bin(Bin).data(1).vectors(end+1) =...
882.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
883.            binPlot(plotnum+1).bin(Bin).data(2).vectors(end+1) =...
884.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
885.            binPlot(plotnum+1).bin(Bin).data(3).vectors =...
886.                [binPlot(plotnum+1).bin(Bin).data(3).vectors, ...
887.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
888.            binPlot(plotnum+1).bin(Bin).data(4).vectors =...
889.                [binPlot(plotnum+1).bin(Bin).data(4).vectors, ...
890.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
891.        end
892.    end
893.end
894.
895.day = norB_sal;
896.plotnum = plotnum+2;
897.for i = 1:4
898.    for b = 1:length(sortByDate(day).ID(i).box)
899.        for Bin = 1:numberOfBins
900.            binPlot(plotnum).bin(Bin).data(1).vectors(end+1) =...
901.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
902.            binPlot(plotnum).bin(Bin).data(2).vectors(end+1) =...
903.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
904.            binPlot(plotnum).bin(Bin).data(3).vectors =...
905.                [binPlot(plotnum).bin(Bin).data(3).vectors, ...
906.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
907.            binPlot(plotnum).bin(Bin).data(4).vectors =...
908.                [binPlot(plotnum).bin(Bin).data(4).vectors, ...
909.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
910.        end
911.    end
912.
913.    for b = 1:length(sortByDate(day+1).ID(i).box)
914.        for Bin = 1:numberOfBins
915.            binPlot(plotnum+1).bin(Bin).data(1).vectors(end+1) =...
916.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
917.            binPlot(plotnum+1).bin(Bin).data(2).vectors(end+1) =...
918.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
919.            binPlot(plotnum+1).bin(Bin).data(3).vectors =...
920.                [binPlot(plotnum+1).bin(Bin).data(3).vectors, ...
921.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
922.            binPlot(plotnum+1).bin(Bin).data(4).vectors =...
923.                [binPlot(plotnum+1).bin(Bin).data(4).vectors, ...
924.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
```

```
925.        end
926.    end
927.end
928.
929.day = all_sal;
930.plotnum = plotnum+2;
931.for i = 1:4
932.    for b = 1:length(sortByDate(day).ID(i).box)
933.        for Bin = 1:numberOfBins
934.            binPlot(plotnum).bin(Bin).data(1).vectors(end+1) =...
935.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
936.            binPlot(plotnum).bin(Bin).data(2).vectors(end+1) =...
937.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
938.            binPlot(plotnum).bin(Bin).data(3).vectors =...
939.                [binPlot(plotnum).bin(Bin).data(3).vectors, ...
940.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
941.            binPlot(plotnum).bin(Bin).data(4).vectors =...
942.                [binPlot(plotnum).bin(Bin).data(4).vectors, ...
943.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
944.        end
945.    end
946.
947.    for b = 1:length(sortByDate(day+1).ID(i).box)
948.        for Bin = 1:numberOfBins
949.            binPlot(plotnum+1).bin(Bin).data(1).vectors(end+1) =...
950.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
951.            binPlot(plotnum+1).bin(Bin).data(2).vectors(end+1) =...
952.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
953.            binPlot(plotnum+1).bin(Bin).data(3).vectors =...
954.                [binPlot(plotnum+1).bin(Bin).data(3).vectors, ...
955.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
956.            binPlot(plotnum+1).bin(Bin).data(4).vectors =...
957.                [binPlot(plotnum+1).bin(Bin).data(4).vectors, ...
958.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
959.        end
960.    end
961.end
962.
963.day = all_sal;
964.plotnum = plotnum+2;
965.for i = 6:8
966.    for b = 1:length(sortByDate(day).ID(i).box)
967.        for Bin = 1:numberOfBins
968.            binPlot(plotnum).bin(Bin).data(1).vectors(end+1) =...
969.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
970.            binPlot(plotnum).bin(Bin).data(2).vectors(end+1) =...
971.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
972.            binPlot(plotnum).bin(Bin).data(3).vectors =...
973.                [binPlot(plotnum).bin(Bin).data(3).vectors, ...
974.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
```

```
975.            binPlot(plotnum).bin(Bin).data(4).vectors =...
976.                [binPlot(plotnum).bin(Bin).data(4).vectors, ...
977.
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
978.        end
979.    end
980.
981.    for b = 1:length(sortByDate(day+1).ID(i).box)
982.        for Bin = 1:numberOfBins
983.            binPlot(plotnum+1).bin(Bin).data(1).vectors(end+1) =...
984.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
985.            binPlot(plotnum+1).bin(Bin).data(2).vectors(end+1) =...
986.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
987.            binPlot(plotnum+1).bin(Bin).data(3).vectors =...
988.                [binPlot(plotnum+1).bin(Bin).data(3).vectors, ...
989.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
990.            binPlot(plotnum+1).bin(Bin).data(4).vectors =...
991.                [binPlot(plotnum+1).bin(Bin).data(4).vectors, ...
992.
    sortByDate(day+1).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
993.        end
994.    end
995.end
996.
997.for k = 1:10
998.    for i = 1:numberOfBins
999.        for j = 1:4
1000.            binPlot(k).bin(i).data(j).mean =
    nanmean(binPlot(k).bin(i).data(j).vectors);
1001.            binPlot(k).bin(i).data(j).std =
    nanstd(binPlot(k).bin(i).data(j).vectors);
1002.            binPlot(k).bin(i).data(j).sem = binPlot(k).bin(i).data(j).std./...
1003.                sqrt(length(binPlot(k).bin(i).data(j).vectors));
1004.        end
1005.    end
1006.end
1007.
1008.%%
1009.%% MORPHINE RESPONSE LATENCIES BINNED
1010.%binPlotMorphine(1) = morphine1, etc
1011.%binPlot(2) = acsf_fss
1012.%binPlot(3) = norB_pre
1013.%binPlot(4) = norB_fss
1014.%binPlot(5) = norB_sal
1015.%binPlot(6) = norB_u50
1016.%binPlot(7) = norB_sal (all)
1017.%binPlot(8) = norB_u50 (all)
1018.%binPlot(9) = acsf_sal (all)
1019.%binPlot(10) = acsf_u50 (all)
1020.
1021.%data(1) = percent correct
1022.%data(2) = number of responses
1023.%data(3) = correct latency
1024.%data(4) = incorrect latency
1025.
1026.for i = 1:numberOfBins
1027.    for j = 1:4
1028.        for k = 1:12
```

```
            binPlotMorphine(k).bin(i).data(j).vectors = [];
        end
    end
end

for day = 83:94
    plotnum = day-82;

    for i = 6:8
        for b = 1:length(sortByDate(day).ID(i).box)
            for Bin = 1:numberOfBins
                binPlotMorphine(plotnum).bin(Bin).data(1).vectors(end+1) =...
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).percentCorrect;
                binPlotMorphine(plotnum).bin(Bin).data(2).vectors(end+1) =...
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).totalNumResponses;
                binPlotMorphine(plotnum).bin(Bin).data(3).vectors =...
                    [binPlotMorphine(plotnum).bin(Bin).data(3).vectors, ...
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).correct_lat];
                binPlotMorphine(plotnum).bin(Bin).data(4).vectors =...
                    [binPlotMorphine(plotnum).bin(Bin).data(4).vectors, ...
    sortByDate(day).ID(i).box(b).latencies.analysis.bins(Bin).incorrect_lat];
            end
        end
    end

end

cor_lat_matrix = zeros(5*4,12);
inc_lat_matrix = zeros(5*4,12);
cor_lat_matrixV = zeros(5*4,12);
inc_lat_matrixV = zeros(5*4,12);

for i = 83:94
    for Bin = 1:4
        for id = 6:8
            for b = 1:length(sortByDate(i).ID(id).box)
                animal = (id-5)*2-2+b;
                cor_lat_matrix(animal*4-4+Bin,i-82) =
    nanmean((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).correct_lat)./100);
                inc_lat_matrix(animal*4-4+Bin,i-82) =
    nanmean((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).incorrect_lat)./100);
                cor_lat_matrixV(animal*4-4+Bin,i-82) =
    var((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).correct_lat)./100);
                inc_lat_matrixV(animal*4-4+Bin,i-82) =
    var((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).incorrect_lat)./100);
            end
        end
    end
end

%Sort by animal
xvalues = {'BL1','BL2','M1','M2','M3','M4','M5','W1','W2','W3','W4','W5'};
yvalues = {'Ani1_15min','Ani1_30min','Ani1_45min','Ani1_60min',...
           'Ani2_15min','Ani2_30min','Ani2_45min','Ani2_60min',...
           'Ani3_15min','Ani3_30min','Ani3_45min','Ani3_60min',...
           'Ani4_15min','Ani4_30min','Ani4_45min','Ani4_60min',...
           'Ani5_15min','Ani5_30min','Ani5_45min','Ani5_60min',};
```

```
figure;
heatmap(xvalues,yvalues,cor_lat_matrix,'ColorLimits',[0 10]);
title('Mean Correct')
figure;
heatmap(xvalues,yvalues,inc_lat_matrix,'ColorLimits',[0 10]);
title('Mean Incorrect')

figure;
heatmap(xvalues,yvalues,cor_lat_matrixV,'ColorLimits',[0,30]);
title('Var Correct')
figure;
heatmap(xvalues,yvalues,inc_lat_matrixV,'ColorLimits',[0,30]);
title('Var Incorrect')

%Sort by bin
for i = 83:94
    for Bin = 1:4
        for id = 6:8
            for b = 1:length(sortByDate(i).ID(id).box)
                animal = (id-5)*2-2+b;
                cor_lat_matrix(Bin*5-5+animal,i-82) =
    nanmean((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).correct_lat)./100);
                inc_lat_matrix(Bin*5-5+animal,i-82) =
    nanmean((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).incorrect_lat)./100);
                cor_lat_matrixV(Bin*5-5+animal,i-82) =
    var((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).correct_lat)./100);
                inc_lat_matrixV(Bin*5-5+animal,i-82) =
    var((sortByDate(i).ID(id).box(b).latencies.analysis.bins(Bin).incorrect_lat)./100);
            end
        end
    end
end

xvalues = {'BL1','BL2','M1','M2','M3','M4','M5','W1','W2','W3','W4','W5'};

yvalues = {'Ani1_15min','Ani2_15min','Ani3_15min','Ani4_15min','Ani5_15min',...
           'Ani1_30min','Ani2_30min','Ani3_30min','Ani4_30min','Ani5_30min'....
           'Ani1_45min','Ani2_45min','Ani3_45min','Ani4_45min','Ani5_45min',...
           'Ani1_60min','Ani2_60min','Ani3_60min','Ani4_60min','Ani5_60min'};

figure;
heatmap(xvalues,yvalues,cor_lat_matrix,'ColorLimits',[0 10]);
title('Mean Correct')
figure;
heatmap(xvalues,yvalues,inc_lat_matrix,'ColorLimits',[0 10]);
title('Mean Incorrect')

figure;
heatmap(xvalues,yvalues,cor_lat_matrixV,'ColorLimits',[0,30]);
title('Var Correct')
figure;
heatmap(xvalues,yvalues,inc_lat_matrixV,'ColorLimits',[0,30]);
title('Var Incorrect')


%% PLOT


BinPlot{1} = 'acsf,pre';
BinPlot{2} = 'acsf,fss';
BinPlot{3} = 'norB,pre';
```

```
1139.BinPlot{4} = 'norB,fss';
1140.BinPlot{5} = 'norB,sal';
1141.BinPlot{6} = 'norB,u50';
1142.BinPlot{7} = 'norB,sal'; %(all)
1143.BinPlot{8} = 'norB,u50'; %(all)
1144.BinPlot{9} = 'acsf,sal'; %(all)
1145.BinPlot{10} = 'acsf,u50'; %(all)
1146.
1147.for i = 1:2:10
1148.    x = bins(2:end);
1149.
1150.    y1 = zeros(4,4);
1151.    s1 = zeros(4,4);
1152.
1153.    y2 = zeros(4,2);
1154.    s2 = zeros(4,2);
1155.
1156.    y3 = zeros(4,2);
1157.    s3 = zeros(4,2);
1158.
1159.    %latencies
1160.    for b = 1:numberOfBins
1161.        row = 1;
1162.        for j = 0:1
1163.            for k = 3:4
1164.                y1(b,row) = [binPlot(i+j).bin(b).data(k).mean];
1165.                s1(b,row) = [binPlot(i+j).bin(b).data(k).sem];
1166.                row = row+1;
1167.            end
1168.        end
1169.    end
1170.
1171.    %percent correct
1172.    for b = 1:numberOfBins
1173.        row = 1;
1174.        for j = 0:1
1175.            for k = 1:1
1176.                y2(b,row) = [binPlot(i+j).bin(b).data(k).mean];
1177.                s2(b,row) = [binPlot(i+j).bin(b).data(k).sem];
1178.                row = row+1;
1179.            end
1180.        end
1181.    end
1182.
1183.    %total responses
1184.    for b = 1:numberOfBins
1185.        row = 1;
1186.        for j = 0:1
1187.            for k = 2:2
1188.                y3(b,row) = [binPlot(i+j).bin(b).data(k).mean];
1189.                s3(b,row) = [binPlot(i+j).bin(b).data(k).sem];
1190.                row = row+1;
1191.            end
1192.        end
1193.    end
1194.
1195.    figure; hold on;
1196.    subplot(3,1,1)
1197.    hbar1 = bar(x,y1); hold on;
1198.    xe1 = cell2mat(get(hbar1,'XData')).'+[hbar1.XOffset];
1199.    errorbar(xe1,y1,s1,'k.');
```

```
1200.    mycolor1=[0 0 1;1 0 0;0 .5 1;1 .5 0];
1201.    colormap(gca,mycolor1)
1202.    xlabel('15 min Bins')
1203.    ylabel('Average Latencies')
1204.    lgd1 = legend(strcat(BinPlot{i}, ',correct'),strcat(BinPlot{i}, ',incorrect'),...
1205.           strcat(BinPlot{i+1}, ',correct'),strcat(BinPlot{i+1}, ',incorrect'));
1206.    lgd1.FontSize = 6;
1207.
1208.    subplot(3,1,2)
1209.    hbar2 = bar(x,y2); hold on;
1210.    xe2 = cell2mat(get(hbar2,'XData')).'+[hbar2.XOffset];
1211.    errorbar(xe2,y2,s2,'k.');
1212.    mycolor2=[0 0 1;0 .5 1];
1213.    colormap(gca,mycolor2)
1214.    xlabel('15 min Bins')
1215.    ylabel('Percent Correct')
1216.    lgd2 = legend(BinPlot{i},BinPlot{i+1});
1217.    lgd2.FontSize = 6;
1218.
1219.    subplot(3,1,3)
1220.    hbar3 = bar(x,y3); hold on;
1221.    xe3 = cell2mat(get(hbar3,'XData')).'+[hbar3.XOffset];
1222.    errorbar(xe3,y3,s3,'k.');
1223.    mycolor3=[0 0 1;0 .5 1];
1224.    colormap(gca,mycolor3)
1225.    xlabel('15 min Bins')
1226.    ylabel('Total # Alternations')
1227.    lgd3 = legend(BinPlot{i},BinPlot{i+1});
1228.    lgd3.FontSize = 6;
1229.end
```

FIBER PHOTOMETRY CODE

LOAD DATA

```
clear all; close all; clc;

global time_gcamp;
global timelin_gcamp;
global sig_gcamp;
global time_hyper;
global timelin_hyper;
global sig_hyper;

subject_1 = 'Subject1-190314-120806-hyperredOnly';
u50_1 = [];

[inject_sig, all_sig] = analyzer(subject_1, u50_1);

savename = strcat(subject_1,'.mat');
save (savename, 'sig_gcamp','timelin_gcamp','sig_hyper','timelin_hyper');
```

ANALYZER FUNCTION

```
function [inject_sig, all_sig] = analyzer(subject, u50)

    global time_gcamp;
    global timelin_gcamp;
    global sig_gcamp;
    global time_hyper;
    global timelin_hyper;
    global sig_hyper;

    %GCAMP
    filename_gcamp = [subject,'_greG.txt'];

    data_gcamp = textread(filename_gcamp,'%s');
    data_gcamp = str2double(data_gcamp);

    [time_gcamp, sig_gcamp] = separate(data_gcamp);

    timelin_gcamp = linspace(0,length(sig_gcamp)/1017, length(sig_gcamp));

    %HYPER
    filename_hyper = [subject,'_redH.txt'];

    data_hyper = textread(filename_hyper,'%s');
    data_hyper = str2double(data_hyper);

    [time_hyper, sig_hyper] = separate(data_hyper);

    timelin_hyper = linspace(0,length(sig_gcamp)/1017,length(sig_hyper));

    %PLOT
    timelin_gcamp_plot = timelin_gcamp(5*1017:end-(5*1017));
    sig_gcamp_plot = sig_gcamp(5*1017:end-(5*1017));

    timelin_hyper_plot = timelin_hyper(5*1017:end-(5*1017));
    sig_hyper_plot = sig_hyper(5*1017:end-(5*1017));

    fig = figure;

```

```
    subplot(2,1,1)
    plot(timelin_gcamp_plot,sig_gcamp_plot, 'k')
    ylabel('GCaMP')
    hold on;
    title(subject)

    subplot(2,1,2)
    plot(timelin_hyper_plot,sig_hyper_plot, 'k')
    ylabel('HyPer')

    inject_sig = [];
    all_sig = [];
end
```

SEPARATE FUNCTION

```
function [time, sig] = separate(data)
    time = [];
    sig = [];
    for i = 1:2:length(data)-1
        time = [time,data(i)];
        sig = [sig,data(i+1)];
    end
End
```

SAMPLE PROGRAM

```
clear all; close all; clc;

sf = 1017; %sampling frequency

data = open('subject-date-time.mat');

sig = data.sig_gcamp;
tim = data.timelin_gcamp;

%% SMOOTH DATA
windowSize = sf*60; %60s smoothing window

%Smooth with window
for j = 1:size(sig,1)
    for i = 1:size(sig,2)-(windowSize-1)
        wind = sig(j,i:i+windowSize-1);
        avg = mean(wind);
        sd = std(wind);
        sig_smoothed(j,i) = avg;
    end
end
time_plot = linspace(0,size(sig_smoothed,2)/sf,size(sig_smoothed,2));

%Smooth with filter
b = (1/windowSize)*ones(1,windowSize);
a = 1;
sig_smoothed = filter(b,a,sig);

%% NORMALIZE DATA TO DELTA F / F
start_baseline = 60;
end_baseline   = 1000;
```

```
baseline = mean(sig_smoothed(start_baseline*sf:end_baseline*sf));

sig_norm = (sig_smoothed-baseline)./baseline;

%% Function Adjust
x = tim(start_baseline*sf:end_baseline*sf);
y = sig_norm(start_baseline*sf:end_baseline*sf);

%Linear Adjust
p = polyfit(x,y,1);

lin_adj = (p(1).*x);

sig_linAdj = sig_norm - lin_adj;

%1st Order Exponential Adjust
f = fit(x,y,'exp1');

coefs = coeffvalues(f);
a = coefs(1);
b = coefs(2);

exp_adj = a*exp(b.*x);

sig_expAdj = sig_norm - exp_adj;

%% Average Across Animals
swim1 = [swim1_1, swim1_2, swim1_3];

swim2 = [swim2_1, swim2_2, swim2_3];

swim3 = [swim3_1, swim3_2, swim3_3];

swim4 = [swim4_1, swim4_2, swim4_3].';

avg_swim1 = mean(mean(swim1));
avg_swim2 = mean(mean(swim2));
avg_swim3 = mean(mean(swim3));
avg_swim4 = mean(mean(swim4));

se_swim1 = std(mean(swim1))./sqrt(size(swim1,2);
se_swim2 = std(mean(swim2))./sqrt(size(swim2,2);
se_swim3 = std(mean(swim3))./sqrt(size(swim3,2);
se_swim4 = std(mean(swim4))./sqrt(size(swim4,2);

%% Fourier Transform
L = length(sig);
Y = fft(sig);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = sf*(0:(L/2))/L;

figure;
plot(f,P1)
ylim([0,1])
xlim([0,30])
xlabel("Swim 1")

%% Average Signal Plot
pelletAll = [pelletSig1, pelletSig2, pelletSig3,...
```

```
94.              pelletSig4, pelletSig5, pelletSig6];
95.
96. pelletAvg = [];
97. pelletStd = [];
98.
99. for i = 1:size(pelletAll,1)
100.    pelletAvg(i) = mean(pelletAll(i,1:end));
101.    pelletStd(i) = std(pelletAll(i,1:end));
102.end
103.
104.figure;
105.plot(pelletTime, pelletAvg,'k-');
106.hold on
107.fill([pelletTime flip(pelletTime)],[pelletAvg-pelletStd
    flip(pelletAvg+pelletStd)],'k','LineStyle','none')
108.hold on
109.alpha(0.25)
110.hold off;
111.
112.
113.%% Bar Plot
114.figure;
115.bar([1,2,3,4],[avg_swim1,avg_swim2,avg_swim3,avg_swim4],0.4,'b');hold on;
116.errorbar([1,2,3,4],[avg_swim1,avg_swim2,avg_swim3,avg_swim4]...
117.                    [se_swim1, se_swim2, se_swim3, se_swim4],'.');hold on;
118.
119.ylim([0,.5])
```

LIVE CELL IMAGING CODE

CELL QUANTIFIER

```
clear all; close all; clc;

% folderPath1 = actualFile folder
folderPath1 = 'C:\Users\Kenzie Marae\Desktop\lif processing\korlite2';
cd(folderPath1); % path of the folder

files = dir('**');
files(end-5:end) = [];
files(1:2) = [];
numfiles = numel(files);
%%
for numits = 1:numfiles

    filename = files(numits).name;
    %StackSize = StackSizes{numits};

    %LOAD FRAMES %%
    [imgHyper, imgMem] = loadFrames(filename); %loadFrames2chan
%     [imgHyper,map] = imread(filename);

    % DEFINE FRAMES TO PROCESS
    for FOI = 1:size(imgHyper,4)

        % FIND CLUSTERS %%
        frameHyper  = imgHyper(:,:,1,FOI);

        [clusterBorders(:,:,FOI)] = findClusters(frameHyper);


        % FILL CLUSTERS%%
        [filledClusters(:,:,FOI)] = fillClusters(clusterBorders(:,:,FOI));

        % WATERSHED CLUSTERS%%
        [filledClusters(:,:,FOI)] = watershedClusters(filledClusters(:,:,FOI));

        % SIZE FILTER AND LABEL CLUSTERS%%
        [labeledImage(:,:,FOI), numberOfClusters(FOI), binaryLabIm(:,:,FOI),
    clusterBorders(:,:,FOI)] = labelClusters(filledClusters(:,:,FOI),
    clusterBorders(:,:,FOI));

        % GET CENTROIDS %%
        [centroids{FOI}] = getCentroids(labeledImage(:,:,FOI));

        % QUANTIFY EACH CLUSTER %%
        [clusterSize{FOI}, avgFluor_Clusters{FOI}] =
    quantifyClusters(labeledImage(:,:,FOI), numberOfClusters(FOI), frameHyper);

    end

    % SORT CLUSTERS INTO DEFINED CELLS %%
    [cellNumbers, maxNumOfCells] = defineCells(centroids);

    % CREATE LABELED IMAGE MASK WITH CELL NUMBERS %%
    [cellLabeledImage, cellBorders] = labelCells(labeledImage, cellNumbers);

    for FOI = 1:size(imgHyper,4)
```

```
54.         % QUANTIFY EACH CELL %%
55.         frameHyper  = imgHyper(:,:,1,FOI);
56. %         frameMem    = imgMem(:,:,1,FOI);
57.
58.         [cellSize{FOI}, avgFluor_Cells{FOI}] = quantifyCells(cellLabeledImage(:,:,FOI),
    maxNumOfCells, frameHyper);
59.     end
60.
61.     % CREATE A MATRIX OF CELL FLUORESCENCE TO PLOT %%
62.     [cellFluorMatrix] = cellFluor2Plot(avgFluor_Cells);
63.
64.     % PLOT CELL FLUORESCENCE OVER FRAMES %%
65.     frames = [1:size(imgHyper,4)];
66.
67.     figure;
68.     ColorSet=varycolor(size(cellFluorMatrix,1));
69.     for i = 1:size(cellFluorMatrix,1)
70.         cellFluorVector = cellFluorMatrix(i,:);
71.         idx = ~any(isnan(cellFluorVector),1);
72.
73.         plot(frames(idx), cellFluorVector(idx),'o-
    ','Color',ColorSet(i,:),'LineWidth',3)
74.         hold on;
75.     end
76.     legend show
77.     title(filename)
78.     xlabel('Time (min)')
79.     ylabel('Cell Fluorescence')
80.
81.     set(gcf, 'PaperUnits', 'inches');
82.     x_width=10; y_width=10;
83.     set(gcf, 'PaperPosition', [0 0 x_width y_width]);
84.
85.     savenameJPG = strcat(filename(1:end-3), 'jpg');
86.     saveas(gcf,savenameJPG)
87.
88.     hold off;
89.
90.     savenameMAT = strcat(filename(1:end-3), 'mat');
91.     saveDirectory = strcat(folderPath1,'\processed\',savenameMAT);
92.     save (saveDirectory, 'cellFluorMatrix',...
93.                          'imgHyper',...
94.                          'cellLabeledImage',...
95.                          'cellBorders',...
96.                          'cellSize');
97.
98.     clearvars -except folderPath1 files numits numfiles
99.
100.end
```

QUANTIFICATION FUNCTIONS

```
1.  %% LOAD FRAMES
2.  function [imgHyper, imgMem] = loadFrames(filename, StackSize)
3.      frames = [3:3:StackSize];
4.      [imgHyper,map] = imread(filename,frames);
5.
6.      frames = [1:3:StackSize];
7.      [imgMem,map] = imread(filename,frames);
8.  end
```

```
9.
10. %% FIND CLUSTERS
11. function [clusterBorders] = findClusters(frameHyper)
12.     frameHypAnalyze = frameHyper; %frameHyper = original frame
13.
14.     %Enhance Bright Regions
15.     thresholdVal = 8;
16.     enhanceAmt   = 25;
17.     for i = 1:size(frameHypAnalyze,1)
18.         for j = 1:size(frameHypAnalyze,2)
19.             if frameHypAnalyze(i,j) >= thresholdVal
20.                 frameHypAnalyze(i,j) = frameHypAnalyze(i,j) + enhanceAmt;
21.             end
22.         end
23.     end
24.
25.     %Preprocess Smoothing and Filtering
26.     sigma     = .1;
27.     alpha     = 5;
28.     numLevels = 10;
29.
30.     frameHypAnalyze = imfill(frameHypAnalyze);
31.     frameHypAnalyze = locallapfilt(frameHypAnalyze,sigma,alpha,'NumIntensityLevels',
    numLevels);
32.
33.     se = strel('disk',3);
34.     frameHypAnalyze = imopen(frameHypAnalyze,se); %frameHypAnalyze = preprocessed
35.
36.     %Find Cluster Borders
37.     clusterBorders = zeros(size(frameHyper,1),size(frameHyper,2));
38.
39.     dist        = 2;
40.     threshUpper = 20;
41.     threshLower = 20;
42.
43.     for r = 1+dist:size(frameHyper,1)-dist
44.         for c = 1+dist:size(frameHyper,2)-dist
45.             p   = frameHypAnalyze(r,c);
46.             pU1  = frameHypAnalyze(r-dist,c);
47.             pL1  = frameHypAnalyze(r,c-dist);
48.             pR1  = frameHypAnalyze(r,c+dist);
49.             pB1  = frameHypAnalyze(r+dist,c);
50.
51.             AOI = [p pU1 pB1 pL1 pR1];
52.
53.             if sum(AOI) ~= 0
54.                 if p >= threshUpper
55.                     if ((pU1<=threshLower & pB1>= threshUpper) | (pU1>=threshUpper &
    pB1<=threshLower))
56.                         clusterBorders(r,c) = 1;
57.                     elseif (pL1<=threshLower & pR1>= threshUpper) | (pL1>=threshUpper &
    pR1<=threshLower)
58.                         clusterBorders(r,c) = 1;
59.                     end
60.                 end
61.             end
62.         end
63.     end
64.
65.     clusterBorders = bwmorph(clusterBorders,'close');
66.     clusterBorders = bwmorph(clusterBorders,'bridge');
```

```
    clusterBorders = bwmorph(clusterBorders,'diag');
    clusterBorders = bwmorph(clusterBorders,'thin'); %bordersClust = borders
end

%% FILL CLUSTERS
function [filledClusters] = fillClusters(clusterBorders)
    inCell = 0;
    last1 = 0;
    numberOfChanges = 0;
    for i = 1:size(clusterBorders,2)
        if clusterBorders(5,i) == 1
            if (inCell == 0) & (last1 == 0)
                inCell = 1;
                last1 = i;
            elseif (inCell == 0) & (last1 == i-1)
                inCell = 0;
                last1 = i;
            elseif (inCell == 1) & (last1 == i-1)
                inCell = 1;
                last1 = i;
            else
                inCell = 0;
                last1 = i;
            end
        end
        if clusterBorders(5,i) == 0
            if numberOfChanges <= 30
                break;
            elseif inCell == 1
                clusterBorders(5,i) = 1;
                numberOfChanges = numberOfChanges + 1;
            end
        end
    end

    filledClusters = imfill(clusterBorders,'holes'); %filledClusters = nonfiltered
clusters
    %figure; imshow(filledClusters)
end

%% WATERSHED CLUSTERS
function [newFilledClusters] = watershedClusters(filledClusters);

    oldClust = filledClusters;
    BW = oldClust;
    D = -bwdist(~BW);
    D(~BW) = -Inf;
    D = imhmin(D,3);
    L = watershed(D);

    for i = 1:size(L,1)
        for j = 1:size(L,2)
            if L(i,j) <= 1
                L(i,j) = 0;
            else
                L(i,j) = 1;
            end
        end
    end

    %figure; imshow(double(L))
```

```
127.
128.    newFilledClusters = logical(L);
129.
130.end
131.
132.%% LABEL CLUSTERS
133.function [labeledImage, numberOfClusters, binaryLabIm, clusterBorders] =
    labelClusters(filledClusters, clusterBorders)
134.
135.    [labeledImage, numberOfRegions] = bwlabel(filledClusters);
136.
137.    minClusterSize = 200; %was 100 for all other quantifications
138.    maxClusterSize = 1500; %was 1000
139.
140.    for i = 1:numberOfRegions
141.        bin = (labeledImage == i);
142.        if sum(sum(bin)) < minClusterSize || sum(sum(bin)) > maxClusterSize
143.            for i = 1:size(labeledImage,1)
144.                for j = 1:size(labeledImage,2)
145.                    if bin(i,j) == 1
146.                        labeledImage(i,j) = 0;
147.                    end
148.                end
149.            end
150.        end
151.    end
152.
153.    [labeledImage, numberOfClusters] = bwlabel(labeledImage);
154.
155.    % CREATE BINARY CLUSTER MASK, REMOVE SMALL CLUSTERS FROM BORDER MAP
156.    binaryLabIm = zeros(size(labeledImage,1),size(labeledImage,2));
157.    for i = 1:size(labeledImage,1)
158.        for j = 1:size(labeledImage,2)
159.            if labeledImage(i,j) ~= 0
160.                binaryLabIm(i,j) = 1; %filtered filled clusters, binary
161.            end
162.%             if labeledImage(i,j) == 0
163.%                 clusterBorders(i,j) = 0; %remove borders surrounding filtered out
    clusters
164.%             end
165.        end
166.    end
167.
168.    %CREATE NEW BORDER MAP
169.    clusterBorders = zeros(size(binaryLabIm,1),size(binaryLabIm,2));
170.    dist        = 1;
171.    threshUpper = 20;
172.    threshLower = 20;
173.
174.    for r = 1+dist:size(binaryLabIm,1)-dist
175.        for c = 1+dist:size(binaryLabIm,2)-dist
176.            p   = binaryLabIm(r,c);
177.            pU1  = binaryLabIm(r-dist,c);
178.            pL1  = binaryLabIm(r,c-dist);
179.            pR1  = binaryLabIm(r,c+dist);
180.            pB1  = binaryLabIm(r+dist,c);
181.
182.            AOI = [p pU1 pB1 pL1 pR1];
183.
184.            if sum(AOI) ~= 0
185.                if p == 1
```

```
                    if ((pU1==0 & pB1==1) | (pU1==1 & pB1==0))
                        clusterBorders(r,c) = 1;
                    elseif (pL1==0 & pR1==1) | (pL1==1 & pR1==0)
                        clusterBorders(r,c) = 1;
                    end
                end
            end
        end
    end

    clusterBorders = bwmorph(clusterBorders,'close');
    clusterBorders = bwmorph(clusterBorders,'bridge');
    clusterBorders = bwmorph(clusterBorders,'diag');
    clusterBorders = bwmorph(clusterBorders,'thin');

end

%% GET CENTROIDS
function [centroids] = getCentroids(labeledImage)
    measurements = regionprops(labeledImage, 'Centroid');

    centroids = zeros(size(measurements,1),2);

    for i = 1:size(measurements,1)
        centroids(i,:) = measurements(i,:).Centroid;
    end
end

%% QUANTIFY CLUSTERS
function [clusterSize, avgFluor_Clusters] = quantifyClusters(labeledImage,
    numberOfClusters, frameHyper);

    %Create labeled cluster masks to break up original frame
    clusterMask = zeros(size(labeledImage,1), size(labeledImage,2), numberOfClusters);

    for k = 1:numberOfClusters
        for i = 1:size(labeledImage,1)
            for j = 1:size(labeledImage,2)
                if labeledImage(i,j) == k
                    clusterMask(i,j,k) = 1;
                end
            end
        end
    end

    %Quantify Each Cluster
    quantifyEachCluster = zeros(size(labeledImage,1), size(labeledImage,2),
    numberOfClusters);
    avgFluor_Clusters = zeros(numberOfClusters,1);
    clusterSize = zeros(numberOfClusters,1);

    for k = 1:numberOfClusters
        quantifyEachCluster(:,:,k) = double(frameHyper).*clusterMask(:,:,k);

        clusterSize(k) = sum(sum(clusterMask(:,:,k)));
        totalFluor  = sum(sum(quantifyEachCluster(:,:,k)));

        avgFluor_Clusters(k) = totalFluor/clusterSize(k);
    end

end
```

```
245.
246.%% DEFINE CELLS
247.function [cellNumbers, clusterNumber] = defineCells(centroids)
248.
249.    %PLOT CENTROIDS OF ALL CLUSTERS ACROSS ALL FRAMES
250.
251.    xVec = [];
252.    yVec = [];
253.    len = [];
254.
255.    for i = 1:size(centroids,2)
256.        a = cell2mat(centroids(i));
257.        x = a(:,1);
258.        y = a(:,2);
259.
260.        len(i) = size(a,1);
261.
262.        xVec = [xVec; x];
263.        yVec = [yVec; y];
264.
265.    end
266.
267.    %hold off;
268.
269.    %CLUSTER CENTROIDS
270.
271.    rng(1);
272.
273.    clusterNumber = max(len); %round(max(len)*1.5); % identify *twice* as many custers
    as the max number of centroids
274.    data = [xVec,yVec]; % Set your data you want to cluster
275.    [idx,C,sumd,D] = kmeans(data, clusterNumber); % idx is the index array, for each
    sample data
276.
277.
278.    %SORT CENTROIDS
279.    [Y,I] = min(D,[],2);
280.
281.    cNew = C;
282.    idxNew = idx;
283.    nullC = [];
284.
285.    frameApp = round(size(centroids,2)/3,0); %cell must appear in atleast 1/3 of
    frames
286.
287.    for i = 1:clusterNumber
288.        a = data(idx==i,:);
289.        if size(a,1) < frameApp
290.            cNew(i,:) = [0,0];
291.            idxNew(idx==i) = 0;
292.            nullC = [nullC,i];
293.        end
294.    end
295.
296.    for i = 1:length(Y)
297.        if Y(i) >= 500
298.            idxNew(i) = 0;
299.        end
300.    end
301.
302.    cellNumbers = {};
```

```
303.    index = 1;
304.    for i = 1:length(len)
305.        cellNumbers{i} = [idxNew(index:index+len(i)-1)];
306.        index = index+len(i);
307.    end
308.
309.end
310.
311.%% LABEL CELLS
312.function [cellLabeledImage, cellBorders] = labelCells(labeledImage, cellNumbers);
313.
314.    cellLabeledImage = labeledImage;
315.
316.    for i = 1:size(cellNumbers,2)
317.        cellLabIm = cellLabeledImage(:,:,i);
318.
319.        cellNums = cell2mat(cellNumbers(i));
320.        numCells = length(cellNums);
321.
322.        cellLabImMask = zeros(size(cellLabIm,1),size(cellLabIm,2));
323.
324.        for j = 1:numCells
325.            cellLabIm_Temp = cellLabIm;
326.            cellLabIm_Temp(cellLabIm_Temp ~= j) = 0;
327.            cellLabIm_Temp(cellLabIm_Temp == j) = cellNums(j);
328.            cellLabImMask = cellLabImMask + cellLabIm_Temp;
329.        end
330.
331.
332.        cellLabeledImage(:,:,i) = cellLabImMask;
333.
334.    end
335.
336.    %CREATE NEW BORDER MAP
337.    cellBorders = zeros(512,512,size(cellLabeledImage,3));
338.    dist        = 1;
339.
340.    for i = 1:size(cellLabeledImage,3)
341.        for r = 1+dist:512-dist
342.            for c = 1+dist:512-dist
343.                p   = cellLabeledImage(r,c,i);
344.                pU1  = cellLabeledImage(r-dist,c,i);
345.                pL1  = cellLabeledImage(r,c-dist,i);
346.                pR1  = cellLabeledImage(r,c+dist,i);
347.                pB1  = cellLabeledImage(r+dist,c,i);
348.
349.                AOI = [p pU1 pB1 pL1 pR1];
350.
351.                if sum(AOI) ~= 0
352.
353.                        if ((pU1<=0 & pB1>=1) | (pU1>=1 & pB1<=0))
354.                            cellBorders(r,c,i) = 1;
355.                        elseif (pL1<=0 & pR1>=1) | (pL1>=1 & pR1<=0)
356.                            cellBorders(r,c,i) = 1;
357.                        end
358.
359.                end
360.            end
361.        end
362.
363.        cellBorders(:,:,i) = bwmorph(cellBorders(:,:,i),'close');
```

```
364.        cellBorders(:,:,i) = bwmorph(cellBorders(:,:,i),'bridge');
365.        cellBorders(:,:,i) = bwmorph(cellBorders(:,:,i),'diag');
366.        %cellBorders(:,:,i) = bwmorph(cellBorders(:,:,i),'thin');
367.
368.    end
369.end
370.
371.%% QUANTIFY CELLS
372.function [cellSize, avgFluor_Cells] = quantifyCells(cellLabeledImage, numberOfCells,
    frameHyper, frameMem);
373.
374.    %Create labeled cluster masks to break up original frame
375.    cellMask = zeros(size(cellLabeledImage,1), size(cellLabeledImage,2),
    numberOfCells);
376.
377.    nullCells = [];
378.    for k = 1:numberOfCells
379.        for i = 1:size(cellLabeledImage,1)
380.            for j = 1:size(cellLabeledImage,2)
381.                if cellLabeledImage(i,j) == k
382.                    cellMask(i,j,k) = 1;
383.                end
384.            end
385.        end
386.        if sum(sum(cellMask(:,:,k))) == 0
387.            nullCells = [nullCells,k];
388.        end
389.    end
390.
391.    %Quantify Each Cell
392.    quantifyEachCell = zeros(size(cellLabeledImage,1), size(cellLabeledImage,2),
    numberOfCells);
393.    avgFluor_Cells = zeros(numberOfCells,1);
394.    cellSize = zeros(numberOfCells,1);
395.
396.    for k = 1:numberOfCells
397.        quantifyEachCell(:,:,k) = double(frameHyper).*cellMask(:,:,k);
398.
399.        cellSize(k) = sum(sum(cellMask(:,:,k)));
400.        totalFluor  = sum(sum(quantifyEachCell(:,:,k)));
401.
402.        avgFluor_Cells(k) = totalFluor/cellSize(k);
403.    end
404.
405.end
406.
407.%% CELL FLOUR 2 PLOT
408.function [cellFluorMatrix] = cellFluor2Plot(avgFluor_Cells)
409.
410.    fluor = cell2mat(avgFluor_Cells);
411.
412.    cellFluorMatrix = [];
413.    for i = 1:size(fluor,1)
414.        if sum(fluor(i,:),'omitnan') ~= 0
415.            cellFluorMatrix = [cellFluorMatrix;fluor(i,:)];
416.        end
417.    end
418.end
```

CELL ANALYSIS

```
clear all; close all; clc;

% folderPath1
folderPath1 = 'C:\Users\Kenzie Marae\Desktop\lif processing\final groups\u50_100nm-vs-veh\u50_100nm';
cd(folderPath1);

files = dir('**');
files(1:2) = [];
numfiles = numel(files);

for numits = 1:numfiles
    filenames_u50{numits} = files(numits).name;
end

% LOAD IN DATA %%
imgHyper_u50 = {};

for i = 1:numfiles
    vars1 = open(filenames_u50{i});
    imgHyper_u50{i} = vars1.imgHyper;
end

%%
% folderPath2
folderPath2 = 'C:\Users\Kenzie Marae\Desktop\lif processing\final groups\u50_100nm-vs-veh\veh';
cd(folderPath2);

files = dir('**');
files(1:2) = [];
numfiles = numel(files);

for numits = 1:numfiles
    filenames_veh{numits} = files(numits).name;
end

% LOAD IN DATA %%
imgHyper_veh = {};

for i = 1:numfiles
    vars2 = open(filenames_veh{i});
    imgHyper_veh{i} = vars2.imgHyper;
end

%%
for i = 1:numfiles
    imgHyper = imgHyper_u50{i};
    for j = 1:size(imgHyper,4)
        frameFluorID(j) = sum(sum(imgHyper(:,:,1,j)>10));
    end

    window = 2;
    frameFluor_runAv = [];
    for m = 1:length(frameFluorID)-(window-1)
        wind = frameFluorID(m:m+window-1);
        avg = nanmean(wind);
        frameFluor_runAv(m) = avg;
    end
```

```
58.
59.     frameFluor_u50{i} = frameFluor_runAv;
60.
61. %     figure;
62. %     plot(frameFluor_runAv)
63.     clear imgHyper frameFluorID frameFluor_runAv
64. end
65.
66. for i = 1:numfiles
67.     imgHyper = imgHyper_veh{i};
68.     for j = 1:size(imgHyper,4)
69.         frameFluorID(j) = sum(sum(imgHyper(:,:,1,j)>10));
70.     end
71.
72.     window = 2;
73.     frameFluor_runAv = [];
74.     for m = 1:length(frameFluorID)-(window-1)
75.         wind = frameFluorID(m:m+window-1);
76.         avg = nanmean(wind);
77.         frameFluor_runAv(m) = avg;
78.     end
79.
80.     frameFluor_veh{i} = frameFluor_runAv;
81.
82. %     figure;
83. %     plot(frameFluor_runAv)
84.     clear imgHyper frameFluorID frameFluor_runAv
85. end
86.
87. %%
88. stackableFrameFluor_u50 = [];
89. for i = 1:length(frameFluor_u50)
90.     frameFluor = frameFluor_u50{i};
91.     baseline = mean(frameFluor(5:9));
92.     frameFluor = (frameFluor - baseline)./baseline;
93.     if size(stackableFrameFluor_u50,2) == 0
94.         diff = 0;
95.     else
96.         diff = size(stackableFrameFluor_u50,2) - size(frameFluor,2);
97.     end
98.     if diff > 0
99.         stackableFrameFluor_u50(i,:) = [frameFluor, zeros(1, diff)];
100.    elseif diff < 0
101.        stackableFrameFluor_u50 = [stackableFrameFluor_u50,
    zeros(size(stackableFrameFluor_u50,1),abs(diff))];
102.        stackableFrameFluor_u50(i,:) = frameFluor;
103.    else
104.    stackableFrameFluor_u50(i,:) = frameFluor;
105.    end
106.end
107.
108.stackableFrameFluor_veh = [];
109.for i = 1:length(frameFluor_veh)
110.    frameFluor = frameFluor_veh{i};
111.    baseline = mean(frameFluor(5:9));
112.    frameFluor = (frameFluor - baseline)./baseline;
113.    if size(stackableFrameFluor_veh,2) == 0
114.        diff = 0;
115.    else
116.        diff = size(stackableFrameFluor_veh,2) - size(frameFluor,2);
117.    end
```

```
    if diff > 0
        stackableFrameFluor_veh(i,:) = [frameFluor, zeros(1, diff)];
    elseif diff < 0
        stackableFrameFluor_veh = [stackableFrameFluor_veh,
   zeros(size(stackableFrameFluor_veh,1),abs(diff))];
        stackableFrameFluor_veh(i,:) = frameFluor;
    else
        stackableFrameFluor_veh(i,:) = frameFluor;
    end
end

%%
mean_u50 = mean(stackableFrameFluor_u50);
sd_u50 = std(stackableFrameFluor_u50);
se_u50 = sd_u50./sqrt(size(stackableFrameFluor_u50,1));

mean_veh = mean(stackableFrameFluor_veh);
sd_veh = std(stackableFrameFluor_veh);
se_veh = sd_veh./sqrt(size(stackableFrameFluor_veh,1));

figure;
plot([1:length(mean_u50)], mean_u50, 'm', 'LineWidth', 3); hold on
plot([1:length(mean_u50)], mean_u50-(se_u50/2), 'm--', 'LineWidth', 3); hold on
plot([1:length(mean_u50)], mean_u50+(se_u50/2), 'm--', 'LineWidth', 3); hold on

plot([1:length(mean_veh)], mean_veh, 'k', 'LineWidth', 3); hold on
plot([1:length(mean_veh)], mean_veh-(se_veh/2), 'k--', 'LineWidth', 3); hold on
plot([1:length(mean_veh)], mean_veh+(se_veh/2), 'k--', 'LineWidth', 3); hold on

plot([1,74],[0,0], 'k', 'LineWidth',3); hold on
plot([11,11],[-.2,.2],'k--'); hold on
plot([60,60],[-.2,.2],'k--'); hold on

xlabel('Frame# (1 Frame / min)')
ylabel('Delta F / F')
title('U50, 100nm vs Veh')
```