Developing and Validating Novel Genetically Encoded Sensors in Animal Behavioral Models to Investigate KOR-Dynorphin Interactions

Brenden Alexander Wong

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Bioengineering

University of Washington

2020

Committee:

Charles Chavkin

Andre Berndt

Program Authorized to Offer Degree:

Bioengineering

University of Washington

Abstract

Developing and Validating Novel Genetically Encoded Sensors in Animal Behavioral Models to Investigate KOR-Dynorphin Interactions

Brenden Alexander Wong

Chair of Supervisory Committee:

Charles Chavkin

Department of Pharmacology

Deaths from the opioid epidemic have been increasing steadily in recent years and have prompted many studies investigating various neural circuits related to addiction. Previous work has shown that both κ-opioid receptor (KOR) agonists and antagonists have potential as therapeutic treatments for individuals suffering from addiction and as alternative analgesics to substitute for addictive substances such as morphine. However, various aspects of KOR-dynorphin actions in different brain regions remain unclear and require further investigation to understand how stress can contribute to substance use disorders. Therefore, we have performed methods for testing, validating, and developing the use of novel genetically encoded sensors to probe KOR-mediated signaling pathways in animal models. We tested these sensors at single cell resolution, bulk neuronal activity, and in behavioral assays. Additionally, we leverage information gained from these sensors to investigate spatial KOR distribution in the PFC and its effect on cognitive disruptions. We found that HyPerRed, kLight, and GCaMP provide useful and unique information regarding circuit dynamics. Ultimately, these findings will allow for more effective investigation into the neural circuits that govern stress and addiction.

Contents

1 Introduction

1.1 Section Overview

The overarching goal of this thesis is to investigate the biological framework for substance use disorders to decrease the danger it poses both to the individual as well as society. Using computational approaches with optogenetic, behavioral, and pharmacological tools, this thesis demonstrates and identifies neural circuits involved with opioid addictions. Furthermore, it demonstrates methods for the development and validation of novel and existing genetically encoded sensors and computational analysis techniques for extracting useful information from those tests. The insights generated from this work will aim to promote therapeutic treatments for substance use disorders.

1.2 Background

The rapid growth of individuals developing opioid addictions has prompted many researchers to seek solutions that will address the opioid crisis. As a result of this problem, there has been an increased toll in terms of economic burden and fatalities placed on our society. It was estimated that the United States spent $78 billion addressing opioid addictions in 2013 (Florence et al., 2013). Additionally, over 100,000 individuals die from opioid overdose annually, and in 2017, over 47,000 deaths were opioid overdose related in the United States (Strang et al., 2020). Abuse of prescription drugs has grown in recent years and it has been estimated that between 21% to 29% of patients misuse prescription opioids and that between 8% to 12% develop an opioid use disorder (Tcherpakov, 2019). Furthermore, prolonged opioid use can produce a variety of unintended consequences, such as tolerance and escalation. A startling statistic estimates that 80% of people who misuse heroin initially began misusing prescription opioids (Tcherpakov, 2019). Orally consumed opioids can also lead to other, more dangerous, forms of drug abuse, including intravenous heroin use (Wikler, 1984). Taken together, these factors have brought dealing with opioid addictions to the forefront of behavioral pharmacology research.

A promising area of interest revolves around finding potential alternative analgesics that are just as effective as morphine, but do not cause dependence. In the case with fentanyl, one proposed mechanism for receptor desensitization is that repeated exposure results in the phosphorylation of opioid receptors, which in turn blocks the receptor from further activity through arrestin recruitment (Alvarez et al., 2002; Allouche et al., 2014; Meiss et al., 2018). This in turn causes the receptor to be internalized and insensitive to opioid ligands. This suggests that drugs that preferentially activate one signaling cascade over the other could be promising targets for developing opioid alternatives. Recent studies suggest that the G protein pathway is responsible for the painkilling effects of opioids while the β-arrestin pathway is responsible for respiratory suppression (Schmid et al., 2017; Kennedy et al., 2018). These G protein biased opioid receptor agonists would be beneficial because they could decouple the cellular response that leads to analgesia from the cellular response that leads to desensitization and eventually displace tolerance inducing analgesics such as morphine.

Furthermore, an additional contributing factor that complicates and reinforces the reward-seeking behavior of drugs through changes in neural circuitry is stress. These same reward-seeking behaviors have different levels of reward based on the stressed state of the individual. Chronic stress is a contributing factor to addiction vulnerability as well as risk of relapse (Goeders, 2003). Therefore, by understanding exactly how stress mediates or contributes to the addiction process, effective treatments could be developed to aid in inhibiting those contributing factors.

1.3 Specific Need

To better understand self-administration of opioids such as morphine in mouse models, we previously developed a tool that was capable of continually monitoring the amount of morphine gelatin consumed by a mouse in a 24-hour period (Abraham et al., 2019, Appendix I). This allowed us to investigate morphine self-administration and withdrawal on a behavioral scale, and how morphine affects broad behavioral trends in mice. However, we lacked the resolution on a physiological level to monitor exactly how morphine affects neural networks and connections to result in morphine dependence and tolerance. Dynorphin, which acts as an agonist for the κ-opioid receptor (KOR) system, is likely released during morphine withdrawal but the timing of such release remains unclear. Therefore, understanding the temporal dynamics of dynorphin release in the brain could aid in developing effective therapeutics for combating addiction in individuals experiencing chronic stressors, such as withdrawal.

In order to effectively identify and develop biased agonists to activate only select pathways that lead to the desired response, it is necessary to first better understand the pathways that result in receptor inactivation and how to characterize and visualize them. One potential mechanism for receptor inactivation includes the generation of reactive oxygen species through NADPH oxidase (Schattauer et al., 2017). Thus, a reporter for reactive oxygen species could serve as an indicator of the activity of the pathway that leads to inactivation of opioid receptors. This study is based on the development and validation of novel sensors that can effectively detect and report the presence of reactive oxygen species or KOR activation in neurons.

1.4 Current Technology

Current technology utilizes other types of sensors such as GCaMP or HyPerRed sensors. These sensors and other similar types will be discussed in future sections. While these sensors provide useful information about different aspects of dynorphin action, other sensors could also help to create a more complete picture regarding the KOR-dynorphin circuit. For example, GCaMP serves as a green fluorescent calcium indicator, which provides information about the release of neurotransmitters at neuronal synapses. However, one of the scientific questions of interest involved monitoring the presence of reactive oxygen species. This suggests that a genetically encoded redox sensor would be more appropriate in this situation, such as HyPerRed. Other tools such as HRM63, a green variant of HyPerRed, and kLight, a sensor indicating KOR activation, could be developed and optimized to highlight specific aspects of the KOR-dynorphin circuit in various brain regions.

1.5 Objectives

This thesis consists of four main objectives that aim to further develop and validate the use of novel sensors in mouse models. At a cellular level, we investigated the response of various cells to pharmacological agents through *ex vivo* single cell imaging and human embryonic kidney (HEK) cells. We had initially planned to include single cell imaging using a head-mounted miniaturized fluorescence microscope (Miniscope) to further develop and validate these sensors. However, due to time constraints and complications arising due to the pandemic, we were unable to assemble this device and we leave this testing as a future direction. To understand broad changes in activity, we used *in vivo* fiber photometry with sensors, consisting of GCaMP, HyPerRed, HRM63, and kLight, to investigate the effects of these pharmacological agents in awake and freely behaving mice. Finally, we investigated the effects these neural circuits have on cognition and motivation for drug seeking through behavioral paradigms. Each of these objectives will be discussed in detail in the following sections.

1.5.1 *Ex vivo* Single Cell Imaging

The first objective for this thesis involves processing and analyzing fluorescence images from *ex vivo* brain slices. These recordings were collected from cells expressing the genetically encoded redox sensors used previously. Single cell recordings provide useful information because we can image many cells at once to visualize how each cell individually reacts to pharmacological treatment. It can provide more information about the time courses of those interactions, which can be difficult using whole field imaging because of destructive interference. Single cell imaging can handle cells that are firing out of phase and extract useful information from those recordings. Therefore, we utilized this method of *ex vivo* single cell imaging to better understand the time course for dopaminergic cells in the VTA reacting to the various pharmacological agents.

For these experiments, I designed an algorithm that used built-in MATLAB functionality to process images generated from slice imaging in single cells *ex vivo*. The custom written script identified certain regions of interest and tracked the fluorescence levels of those cells over the time course of the experiment, as well as tracked the fluorescence levels of the whole field. This provided more information about how the fluorescent sensors in these cells respond to certain pharmacological agents and provided a more complete picture about the average time course of those drugs in these specific cell types in the VTA.

Some disadvantages to this method include the inability to link responses at a cellular level to behavior. Since these experiments are performed *ex vivo*, it is difficult to determine how those responses would differ if they had occurred in a living system (*in vivo*). Therefore, it was necessary to additionally test the pharmacological agents *in vivo* as well, using the fluorescent sensors to understand those interactions.

1.5.2 Single HEK Cell Imaging

Secondly, we processed and analyzed fluorescence images from HEK cells expressing genetically encoded redox sensors, such as HyPerRed and HRM63, that were treated with various pharmacological agents targeting opioid receptors, such as nalfurafine or U50,488. While HEK cells and neuronal cells have different compositions, these cells are useful tools for rapid initial testing of the redox sensors mentioned previously. Additionally, testing with HEK cells is more cost effective than performing analogous experiments in the brain. As such, these tests were useful to first verify the functionality of the sensors. However, since the HEK cells have different compositions than neurons, it was unknown whether these sensors would be useful in a neuronal context.

For these experiments, I adapted and further optimized an algorithm previously used to identify regions of interest in images of fluorescent HEK cells expressing sensor proteins (Andrews, 2019). The cell regions were identified and tracked throughout the course of the recording and individual changes in fluorescence for each cell were recorded. Through these cellular resolution fluorescence data, we were able to accurately determine the time course over which various pharmacological agents produced their effect.

1.5.3 *In vivo* Fiber Photometry

To understand bulk neural activity, we utilized *in vivo* fiber photometry to monitor how pharmacological agents affected neural activity in specified brain regions. For these experiments, we used either the sensors GCaMP6m, HyPerRed, HRM63, or kLight1.3. GCaMP is a well-established sensor that serves as an indicator for the presence of calcium, which is released in active neurons. HyPerRed is a sensor that indicates the presence of reactive oxygen species, which are produced in a specific pathway following activation of opioid receptors. HRM63 is a variant of HyPerRed that fluoresces at a green-shifted wavelength, as opposed to HyPerRed, which fluoresces at a red-shifted wavelength. kLight is a green fluorescent sensor that fluoresces upon dynorphin binding and thus serves as a direct measure for dynorphin activity.

Validation of these sensors *in vivo* was a necessary step to confirm the results obtained from the single cell imaging experiments. These experiments provided information about when groups of neurons in certain brain regions were active and about relevant time courses that could affect interactions within the brain, but they lacked the cellular resolution that was present in the previous analyses. When paired with behavioral paradigms, they provided information to connect bulk neural activity with observed behavior in the mice, which is something that was not possible with the *ex vivo* single slice imaging or the single HEK cell imaging.

1.5.4 Behavioral Analysis

Finally, to investigate broad behavioral effects of certain drugs, we ran various behavioral assays under different conditions. We performed conditioned place preference experiments with repeated force swim stress to model stressful situations and the rewarding effects of drugs. Additionally, we used an operant delayed alternation paradigm that investigated the working memory of mice in performing a task for a food pellet reward under various conditions. These behavioral experiments provided information about how certain pharmacological agents affect behavior of the mice through drug seeking behaviors under stress as well as effects on working memory.

1.6 Conclusions

Taken together, the information gained from utilizing computational techniques for analysis of novel genetically encoded sensors provided useful information at various resolutions, including cellular, neural circuit, and behavioral levels. The insights gained from this thesis provide a more detailed understanding of the possibility of biased agonists and agonists as addiction therapeutics as well as a better understanding of how stress affects addiction circuits. Moreover, the insights also elucidate further the role of potential treatments on behavior such as drug seeking motivation and cognition. We were able to accomplish this by beginning at a cellular resolution and increasing the scope to focus on increasingly broader systems, ultimately leading to behavioral outputs. This process is summarized in **Figure 1**.

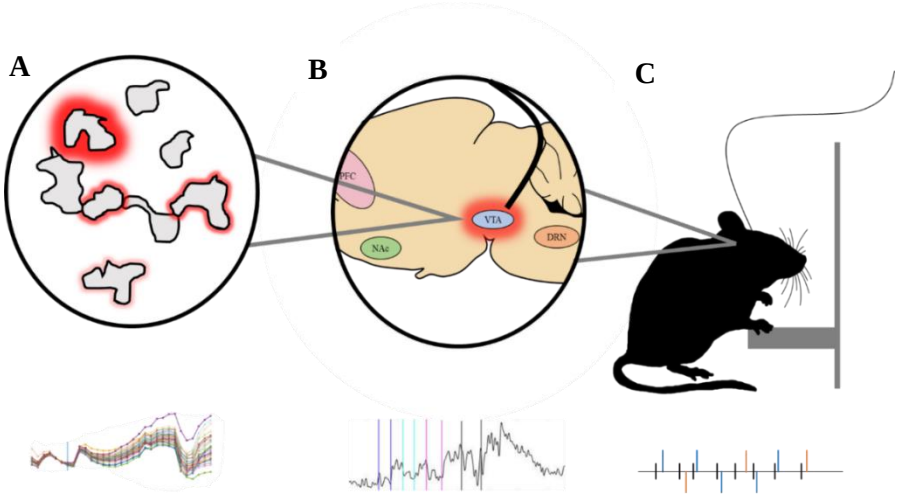


Figure 1: Analyses at varying resolutions of neuronal dynamics can determine the efficacy of different genetically encoded sensors. *These analyses inform on which sensors could provide useful information at these levels. (A) Experiments were performed at a single cell resolution on ex vivo brain slices and HEK cells to test various genetically encoded sensors. (B) Sensors were then tested and validated on a bulk population level with fiber photometry to verify feasibility for behavioral experiments. (C) Mice were then treated with pharmacological agents and stressors to observe the effect on behavior and link insights gained from the previous validation steps with behavioral observations.*

2 Literature Review

2.1 Section Overview

In this section, we provide an analysis on current literature investigating the fundamental neural circuits and connections involved in addiction, the contributing effects of stress on addiction, relevant pharmacological agents involved in investigating these effects, and the specific brain regions being targeted. Additionally, we provide an analysis on the optogenetic tools and methods currently being used to investigate these interactions.

This section provides the theoretical basis for understanding the methods and motivation for performing the trials and experiments that will be described hereafter.

2.2 Physiological Basis of Addiction

Drug addiction has been characterized by consisting of three phases: drug binging and intoxication, withdrawal and negative affect, and preoccupation and craving (Koob and Volkow, 2009). For each of these phases, specific brain regions, cell types, and circuits are all involved that result in these behaviors.

Essentially all addictive drugs lead to an increase in the neurotransmitter dopamine release, which acts as a reinforcement signal (Koob, 1992; Chiara, 2002; Wise, 2008). By pairing rewards with certain environmental contexts, dopamine levels will increase when presented with the conditioned context, even without administration of the addictive drug. Under normal circumstances, certain aspects of individuals' lives provide natural rewards, such as food, water, and sex. However, these drugs of abuse can take over natural reward signaling pathways and replace those signals with signals resulting from drug use and the stimuli associated with it (Olsen, 2011; Volkow et al., 2016). By suppressing dopamine levels through drug use, this also affects the prefrontal cortex in the brain, which is responsible for decision making processes (Volkow et al., 2016). This change in dopamine

regulation causes behavioral changes linked with the prefrontal cortex, and ultimately changes how the individual responds to strong urges.

As drug use induces changes in neuroplasticity and alters the reward pathways in the brain which ultimately drive individuals to seek drugs of abuse, it also alters stress pathways in the brain. The adaptations also amplify the effects of stress on the individual (Davis et al., 2009; Jennings et al., 2013; Volkow et al., 2016). As a result, the individual experiences motivation, not only to satisfy a desire for the rewarding effects of drug use, but also to escape dysphoric stress induced situations, thereby strengthening the drive to seek drugs of abuse.

Therefore, there are critical points in the phases of addiction that could break the addiction cycle and be targeted in order to develop treatments or preventions for addiction: the positive reinforcement during binge and intoxication, the negative reinforcement during withdrawal and negative affect, and the behavioral changes linked with preoccupation and craving. By either reducing the level of dysphoria induced by stressful situations or by reducing the rewarding effects of drug use, therapeutics that target these aspects could help individuals to overcome substance use disorders.

2.3 Effect of Stress on Addiction

Stress is a natural response that is necessary for survival of all species. While acute stress can be beneficial and constructive for individuals, chronic stress has been shown to negatively impact mental states, fundamental biological functions, and neural circuitry. Over time, this results in an allostatic load (McEwen, 2011; Gold et al., 2015). These changes can affect nearly all facets of an individual's life and can be especially potent in individuals with substance use disorders.

For example, while stress associated hormones help the body to adapt in response to acute stressors through allostasis, chronic stress can result in alterations that can alter the connections and structure of key brain regions and thereby modify the body's behavioral response (McEwen, 2011). To further illustrate the difference in behavioral responses in acutely versus chronically stressed subjects, open field tests with rats have shown that acute exposure to stressors resulted in increased open field activity, whereas rats with chronic exposure to stress had decreased open field activity from baseline levels (Katz et al., 1981). Additionally, chronic stress has been shown to affect spatial memory and learning (Conrad, 2010). Thus, chronic stress appears to distinctly impact individuals as compared with acute stress.

An important component of the body's stress response system is the hypothalamic-pituitary-adrenal (HPA) axis. This endocrine system controls the body's response to stress and affects brain regions also tied with learning and memory, such as the medial prefrontal cortex, the hippocampus, and the amygdala (Green and McCormick, 2013). As a result, chronic stressors not only have an impact on the physiological functions and mental state of individuals as discussed previously, but they also modify neural circuits and brain structures to affect learning and memory in chronically stressed individuals.

Physiologically, some of the key components in stress circuits are corticotropin-releasing factor (CRF) and dynorphin (Nestler and Carlezon, 2006). It has been well studied that CRF is a primary messenger that allows the central nervous system to control the HPA axis (Owens and Nemeroff,

1991). CRF acts as an agonist for both CRF_1-receptors and CRF_2-receptors, which has a combined effect that leads to a depressive state. As such, CRF-receptor antagonists are commonly prescribed as anti-depressants in humans (Land et al., 2008).

Sustained stressful situations can have important implications on the behavioral response of individuals responding to drugs of abuse and can contribute to the withdrawal and negative affect stage that was mentioned previously by increasing the risk of relapse. It has been well documented that, along with the behavioral changes that were mentioned previously, chronic stress can also lead to an increase in risk of drug abuse (Volkow and Li, 2004; Koob and Kreek, 2007). Additionally, by exposing animals to constant and inescapable stressors, there is an increase in the rewarding properties of drugs as well as the drive for self-administration (Piazza et al., 1990; Shaham and Stewart, 1994; McLaughlin et al., 2003). As such, stress and its effects on neural circuits are a major focus of research as well as a target for developing effective treatment options for people faced with substance use disorders.

2.4 Role of the κ-Opioid Receptor and Dynorphin

Besides CRF, one of the other critical components of stress circuits that contribute towards addiction is the interaction between dynorphin and the κ-opioid receptor (KOR). KORs are seven-member transmembrane G protein coupled receptors (GPCRs) and have been implicated as possible targets for treating drug addictions. Like other opioid receptors, activation of KOR can result in an analgesic response, but unlike other opioid receptors, it can also lead to a dysphoric effect, thereby limiting its use as a therapeutic target (Carlezon et al., 1998, Bruchas and Chavkin, 2010, White et al., 2013; Tejeda et al., 2013; Ehrich et al., 2015). However, it has still been shown to be an effective target for treating cocaine dependence and self-administration (Glick et al., 1995; Bidlack, 2014). Therefore, finding a way to disentangle pathways activated by KOR and activate select desirable pathways could prove to be an effective method for developing treatment options for those with substance use disorder. This search for biased KOR agonists makes it a promising target for treatment options.

One of the agonists that activate KOR are endogenous peptides such as dynorphin (Bruchas and Chavkin, 2010). It was found that dynorphins, which are derived from prodynorphin, produce different effects when binding to KOR compared with the other opioid receptors (Chavkin et al., 1982). Additionally, while dynorphins can serve as an agonist and bind to any of the three opioid receptors (δ-, μ-, and κ-opioid receptors), it shows preference towards KOR. In many cases, however, it can activate opposite motivational effects when binding to KOR than with the μ-opioid receptors. As discussed previously, dynorphins and other KOR agonists produce dysphoric effects in both animals and in humans, while μ-opioid receptor agonists such as morphine or heroin produce euphoric effects (Shippenberg et al., 2007; Koob et al., 2014). In general, it is fairly well understood that agonist binding to opioid receptors leads to the desensitization of those GPCRs. For KOR, dynorphin is one such ligand that causes the desensitization and internalization through phosphorylation mechanisms (Bruchas and Chavkin, 2010). Thus, dynorphin emerges as an important agonist that contributes to opioid tolerance.

Contrasting with the use of agonists, treatment with KOR antagonists have shown to be useful in mitigating the stress response in stressed animals and attenuate the prodepressive-like responses in those animals (Newton et al., 2002; Knoll and Carlezon, 2010). Since stress can be a contributing factor to drug addiction, blocking the effects of endogenous stress induced KOR agonists such as dynorphin could be beneficial as treatment options for addiction, suggesting the use of KOR antagonists.

2.4.1 KOR Expression

KOR has high expression in the ventral tegmental area (VTA), the hippocampus (HP), the prefrontal cortex (PFC), the nucleus accumbens (NAc), the bed nucleus of the stria terminalis (BNST), the amygdala (AMY), and the dorsal raphe nucleus (DRN), which are all mesocorticolimbic brain areas (Swanson et al., 1982). The VTA contains projections to other brain regions in the HP, PFC, NAc, BNST, and AMY, while the DRN also contains projections to the NAc. All these connections are KOR mediated and can induce changes in behavior in a KOR dependent manner. The important effects of these brain regions are summarized below in **Table 1**, adapted from Veer and Carlezon.

Table 1: Neural circuits expressing KOR. *Certain brain regions expressing KOR have important effects on behavior related to drug addiction. Understanding the temporal and spatial dynamics of these circuits can provide insights as to how to effectively reduce threats of drugs of abuse.*

Brain Region	Expresses KOR?	Projections	Effects
Ventral Tegmental Area	Yes	Outputs to HP, PFC, NAc, BNST, AMY	Reward signaling
Hippocampus	No	Inputs from VTA	Memory, decision making
Prefrontal Cortex	Yes	Inputs from VTA	Cognition, decision making
Nucleus Accumbens	Yes	Inputs from VTA	Aversion/reward processing
Bed Nucleus of the Stria Terminalis	Yes	Inputs from AMY	Anxiety, reinstatement
Amygdala	No	Inputs from VTA, outputs to BNST	Withdrawal and relapse, anxiety
Dorsal Raphe Nucleus	Yes	Inputs from NAc	Withdrawal and depression

2.5 KOR Signaling Cascades

As mentioned previously, KOR is a seven-member transmembrane GPCR with subunits Gα and Gβγ and is expressed in specific brain regions that can affect various behavioral aspects, such as cognition, stress, and anxiety (Veer and Carlezon, 2013). Upon KOR activation by agonists such as dynorphin or U50,488, multiple kinase cascades become activated (Bruchas and Chavkin, 2010). Binding of a KOR agonist causes the exchange of a bound GDP on the Gα subunit with a GTP, which in turn causes the dissociation of the Gα and Gβγ subunits from the receptor (Wettschureck and Offermanns, 2005). The dissociation of these subunits from each other and from the receptor

are key steps in the signaling cascade that results from ligand binding to KOR that act as effectors for other chemical species of interest.

2.5.1 The G Protein Pathway

The G protein pathway is thought to be responsible for the analgesic effects of KOR activation (Schattauer et al., 2017a). After agonist binding, the Gβγ and Gα subunits dissociate, as described previously, and the Gβγ subunit goes on to activate G protein gated inwardly rectifying potassium channel as well as inhibits the influx of calcium through voltage-gated channels (Rusin et al., 1997; Bruchas and Chavkin, 2010; Schattauer et al., 2017a). As a result, the membrane becomes hyperpolarized, thereby decreasing the likelihood of firing an action potential. Additionally, the Gβγ subunit is responsible for the early phase extracellular signal-regulated kinases 1 and 2 (ERK1/2) activation (McLennan et al., 2008).

Like other GPCRs, the Gα subunit of KOR regulates the levels of cyclic adenosine monophosphate (cAMP). Specifically, it was found that the Gα subunit of activated opioid receptors inhibits production of cAMP (Minneman and Iversen, 1976).

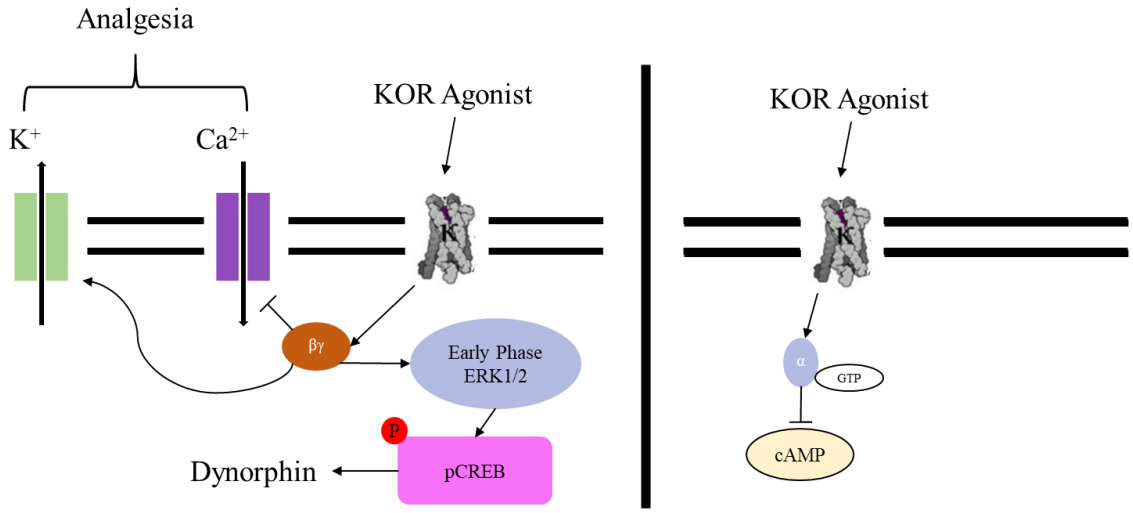


Figure 2: Activation of KOR results in the separation of the Gβγ and Gα subunits from the receptor. *(Left) The Gβγ subunit causes a reduction of influx of calcium into the cell while opening potassium channels for potassium efflux from the cell. Additionally, it activates ERK1/2 during the early phase of the signaling cascade which then phosphorylates pCREB, ultimately resulting in the expression of dynorphin. (Right) One effect of dissociation of the Gα subunit from the receptor is the inhibition of cAMP production.*

2.5.2 The GRK3/Arrestin Pathway

The arrestin-mediated KOR activated pathway has been proposed to be responsible for the main dysphoric effects that result from agonist binding (Bruchas et al., 2007; White et al., 2013).To initiate arrestin-dependent signaling, GRK3 is responsible for receptor desensitization and

internalization of KOR (Bruchas and Chavkin, 2010). After the Gα and Gβγ subunits dissociate from the receptor, GRK3 can then phosphorylate the receptor and lead to arrestin recruitment. This phosphorylated GPCR-arrestin complex can then go on to recruit different MAPK signaling cassettes, which is responsible for activating various signaling pathways (Lefkowitz and Shenoy, 2005).

Some of the key MAPKs that are recruited are ERK1/2 and p38 MAPK. While the G protein pathways are responsible for the early phase ERK1/2 activation, the arrestin-dependent pathway is responsible for the late phase activation (McLennan et al., 2008). Through KOR-mediated ERK1/2 activation, pCREB is phosphorylated, which regulates the dynorphin gene expression (Carlezon et al., 1998). While ERK1/2 activation can occur through both arrestin-dependent and arrestin-independent mechanisms, p38 MAPK depends solely on the GRK3/arrestin complex (Bruchas et al., 2006). The p38 MAPK has important implications relating to KOR-mediated aversion (Bruchas et al., 2007).

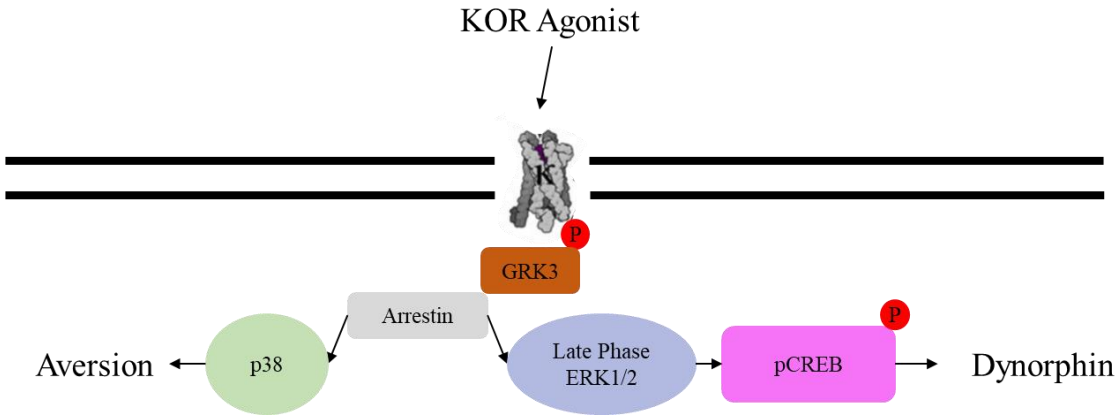
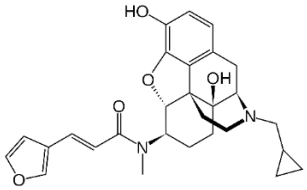
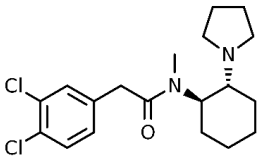
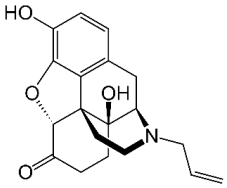
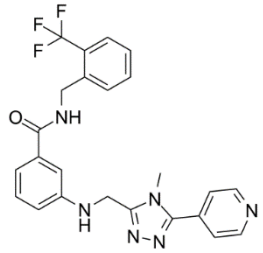
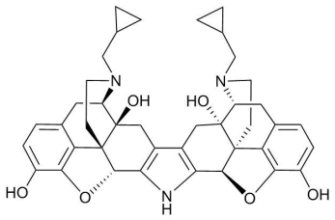
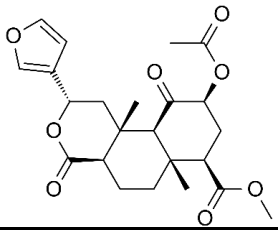


Figure 3: KOR activation additionally results in activation of the GRK3/Arrestin pathway. *After the Gβγ and Gα subunits dissociate from the receptor, GRK3 can bind and recruit arrestin. This recruitment then goes on to activate ERK1/2 in the late phase of the cascade, which goes on to phosphorylate pCREB and result in the expression of dynorphin. Additionally, arrestin recruitment results in the activation of p38 MAPK, which in turn is responsible for the aversive components of KOR activation.*

2.6 Pharmacological Agents

To investigate the KOR-dynorphin system and efficacy of certain sensors, we used various pharmacological agents to enact physiological changes in key components of the KOR signaling cascades. These pharmacological agents consisted of various biased and unbiased KOR agonists, selective and nonselective opioid receptor antagonists, and inhibitors of key components of the KOR signaling cascades. The compounds, their principal activities, and their specific effects are listed below in **Table 2**.

Table 2: Summary of pharmacological agents used throughout this thesis. *Pharmacological agents used range from biased or unbiased KOR agonists, nonselective and selective opioid receptor antagonists, and pathway inhibitors.*

Name of Compound	Principal Activity	Effect
Nalfurafine 	Biased KOR agonist	Biased KOR agonist that activates the arrestin-independent pathway, leading to analgesia without dysphoria
U50,488 	Unbiased KOR agonist	Activates KOR-mediated arrestin-independent and arrestin-dependent pathways, leading to both analgesia and dysphoria
Naloxone 	MOR/KOR antagonist	Acts as nonselective opioid antagonist to block signaling pathways
CMPD101 	GRK2/3 inhibitor	Prevents arrestin-binding to KOR, and thus activation of arrestin-dependent pathway, without affecting the arrestin-independent pathway
Norbinaltorphimine 	KOR antagonist	Acts as a long lasting KOR selective antagonist to block KOR signaling pathways
Salvinorin A 	Biased KOR agonist	KOR agonist whose bias level is unclear.

2.7 Reactive Oxygen Species in GPCR Inactivation

One of the greatest contributing factors to opioid addictions is the development of tolerance. This tolerance stems from opioid receptors becoming inactivated and results in higher dosages of opioid necessary to elicit the same effect. It is still not fully understood which cellular pathways and receptors are responsible for causing this tolerance and resulting dependence on opioids. However, there have been several signaling pathways and receptors that have been proposed as contributing factors to opioid desensitization and tolerance. One of which involves the depalmitoylation of the Gαi subunit of KOR through oxidation, which is induced by the cJun N-terminal Kinase (JNK) dependent peroxiredoxin 6 (PRDX6) recruitment and production of reactive oxygen species (ROS) through increased NADPH Oxidase (NOX) activity (Schattauer et al., 2017b). These reactive oxygen species could also diffuse to other local receptors and inactivate their respective Gαi subunits (**Figure 4**).

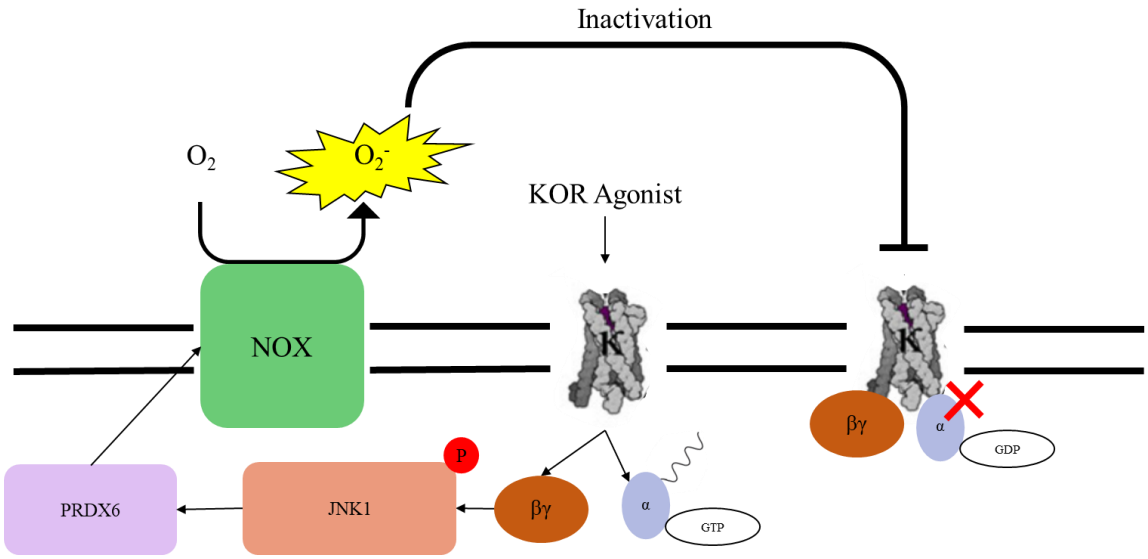


Figure 4: KOR activation results in the dissociation of the Gβγ and Gα subunits. *In addition to the effects previously mentioned, the Gβγ subunit also phosphorylates JNK1, which activates PRDX6 which goes on to activate NOX. Activation of NOX results in the generation of reactive oxygen species, which induces depalmitoylation of the Gβ subunit and inactivates KOR. The reactive oxygen species can be detected by the reactive oxygen species sensors developed in this thesis.*

Detection of ROS is important for determining activity of KOR through JNK-dependent PRDX6 recruitment and for tracking KOR inactivation. For this reason, we have adapted and developed genetically encoded redox sensors to track postsynaptic KOR activation through the G protein pathway in mouse models. By appropriately detecting spikes in the concentration of reactive oxygen species, this enables us to determine the extent of KOR activity through this arrestin-independent pathway.

Additionally, KOR agonists that are G protein biased such as nalfurafine could have important implications for opioid replacements. By preferentially activating the G protein pathway of KOR

signaling, it can induce the same analgesic effects of opioids, but has low activation of the dysphoric component that results from the arrestin-dependent pathway (Schattauer et al., 2017a). As a result, it is useful to have a sensor that can detect and can be used to quantify reactive oxygen species levels in a physiological environment.

2.8 Sensors

In the field of cellular imaging, fluorescent sensors have increasingly been used to characterize molecular interactions that take place within the cell that would be otherwise nearly impossible to visualize. Previous fluorophores that have been used to visualize cellular events include fluorescein isothiocyanate (FITC; The and Felkamp, 1970), fluorescence resonance energy transfer (FRET) based sensors (Marx 2017), and circularly permuted fluorescent proteins (Topell et al., 1999). These sensors provided the technological basis that allowed researchers to study neural circuitry, both *ex vivo* and *in vivo*. Additionally, there are many ways to analyze signals collected from these types of sensors, which can provide useful information when interpreting the overall physiological effects.

Since the chemical interactions that lead to addiction and tolerance occur on a microscopic level and cannot be readily visualized without specialized techniques, the use of indicators for chemical species is of great intrinsic value. The fluorescent nature of these probes allows for visualization of the concentration of the chemical species of interest under certain conditions, which is correlated with the intensity of the fluorescence.

There are many different methodologies that employ various configurations of sensors to exploit the fluorescent nature of these probes. In addition, the molecular composition itself of the probe can alter the specific wavelength of light absorbed and transmitted. Thus, the selection of an appropriate sensor is modular and results in a broad array of possibilities, each suitable for specific types of experiments. Furthermore, fluorescence from these sensors can then be processed and analyzed through a variety of methods to extract useful information from the model. As a result, there are many combinations of available sensors and analysis techniques that can be selected from to best suit the experimental procedure and provide the most useful information. The different types of sensors will be described in the following sections.

2.8.1 Types of Sensors

The initial class of fluorescent probes that was developed were small molecule probes. One of the early uses of these probes was with the conjugation of a small fluorescent molecule, such as anthracene, with an antibody, such as Pn 3 serum (Coons et al., 1942). When the antibody-tracer conjugate was exposed to ultraviolet light, it emitted a blue fluorescence as a response. However, it was noted that this emission spectrum overlapped significantly with the inherent emission spectra of the tissue in question, rendering it more difficult to observe fluorescence solely due to the

conjugate. Since then, various techniques have been developed to improve the sensitivity and specificity of sensors.

FRET based sensors work by using absorption and emission characteristics of the sensors. The system consists of two fluorophores, each that absorb and emit light at different wavelengths. In isolation, the first sensor absorbs at the wavelength λ_1 and emits light at the wavelength λ_2. When the two sensors are in close proximity, the first sensor absorbs at λ_1, emits light at λ_2 which is then absorbed by the second sensor, which then emits light at the wavelength λ_3 (Vinkenborg et al., 2009; Hamers et al., 2013). The system can be used for a wide variety of tests, including identification of recruitment to membranes as well as with linkers conjugated between that change conformational shape in the presence of messengers such as calcium.

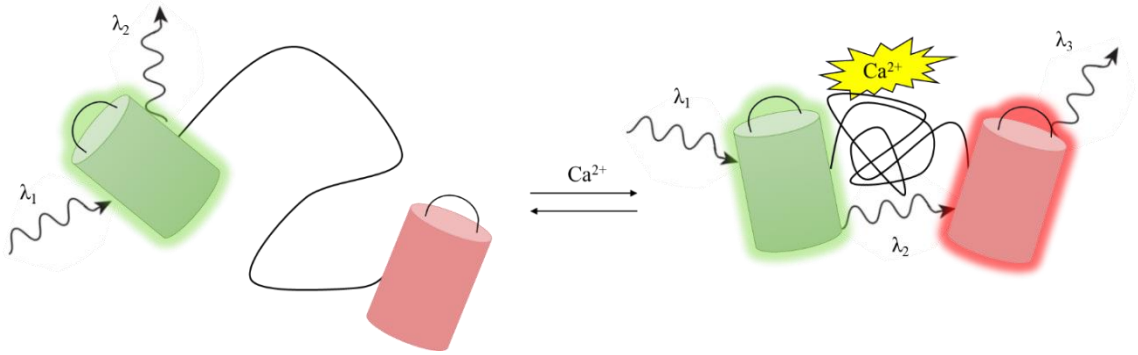


Figure 5: FRET sensors allow for detection of chemical species of interest based on the proximity of two fluorophores. *(Left) In the absence of the chemical species of interest, a certain wavelength of light within the absorbance spectrum of the first fluorophore is absorbed, and a different wavelength is transmitted that is within the emission spectrum of the first fluorophore. (Right) In the presence of the chemical species of interest, the linker between the two fluorophore undergoes a conformational change that brings them closer together. When the first wavelength of light is absorbed by the first fluorophore, it emits a second wavelength that is then absorbed by the second fluorophore. The second fluorophore then emits a third wavelength of light, and detection of this wavelength determines the presence of the chemical species of interest.*

Circularly permuted fluorescent proteins are based on Green Fluorescent Protein (GFP). GFP was originally isolated from the jelly fish *Aequorea victoria* and has a structure comprised of 11 β strands that form a barrel (β-barrel), an α-helix that runs up the center of the barrel, and a chromophore attached to the α-helix in the center of the barrel (Tsien, 1998). The cpGFP is created by interchanging the carboxyl and amino termini through fusing the original termini with a linker and breaking the protein in the β-barrel (Nagai et al., 2001). A sensing domain is fused to the new carboxyl and amino termini that is used to detect the chemical species of interest. When the domain is unbound, water can permeate the barrel and quench the fluorophore in the center. When the domain is bound, it causes a conformational change in the barrel and water is unable to permeate, causing the fluorophore to emit and indicate the presence of the species of interest.

These types of sensors are flexible in their utility, since many different sensing domains can be combined with various types of reporter domains that fluoresce at different wavelengths with variable linker sizes between the two domains. However, designing such a sensor requires optimization to achieve acceptable sensitivity and fluorescence levels, which can be the most time-consuming step when developing new sensors.

2.8.2 Genetically Encoded Calcium Indicators

Calcium is a critical species present in physiological conditions that provides useful information about neuronal activity. Calcium ions are crucial components for the firing of action potentials as well as synaptic inputs. Since they are ubiquitous for all neuronal cells, they can provide useful information about action potential firing through the measurement of calcium ion concentration (Yasuda et al., 2004). As a result, genetically encoded calcium indicators were developed for this purpose. Genetically encoded calcium sensors are classified as circularly permuted fluorescent proteins that have a calmodulin fused to the C terminus and an M13 (a fragment of the myosin light chain kinase) fused to the N terminus (Nakai et al., 2001). In the presence of Ca^{2+}, CaM becomes bound and undergoes a conformational change, allowing for binding with the M13 domain (Wang et al., 2008). The Ca^{2+}-CaM-M13 complex causes the conformational change in the β-barrel as described above and causes fluorescence. Thus, fluorescence can be used as a reporter for the concentration of Ca^{2+} and thereby indicates neuronal activity.

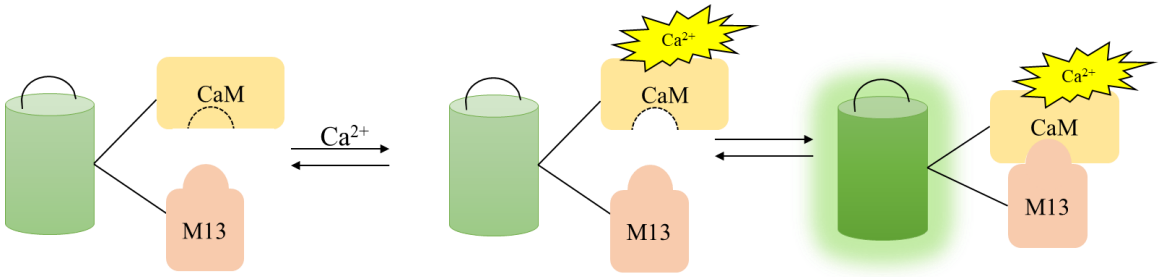


Figure 6: GCaMP utilizes calmodulin and the M13 domain to report the presence of calcium. *When calcium is absent, M13 and CaM do not bind. In the presence of calcium, CaM undergoes a conformational change that allows M13 to bind. When the two units bind, this results in a conformational change in the β-barrel that prevents water from quenching the fluorescence. As a result, in the presence of calcium, this sensor fluoresces.*

2.8.3 Genetically Encoded Redox Sensors

While calcium serves as a second messenger in many signaling cascades and is traditionally utilized for monitoring neural activity, other indicator species could be of more interest for certain applications, such as reactive oxygen species. Probes utilizing interactions between reactive oxygen species and boronate groups have been used previously, however there are complications with quantitative analyses of signals resulting from such sensors (Woolley et al., 2013). Another solution is to use a reactive oxygen species sensing domain in place of the CaM-M13 binding domains used in genetically encoded calcium indicators such as GCaMP. Various sensors such as HyPer (Belousov et al., 2006) and roGFP2-ORP1 (Gutscher et al., 2009) have been designed to report the presence of reactive oxygen species. These sensors are green fluorescent and have a significant spectral overlap with other common sensors such as GCaMP that precludes their use in multiplexing experiments that would require both calcium and reactive oxygen species sensors. As a result, a novel red fluorescent genetically encoded redox sensor was developed to allow for such experiments.

The HyPerRed sensor is similar to GCaMP and other such sensors in that it consists of a sensing domain and a reporter domain. The reporter domain is a variation of cpGFP, where the β-barrel is circularly permuted mApple, or cpmApple (Ermakova et al., 2014). Since mApple is used in the place of GFP, HyPerRed is a red-fluorescent sensor. The sensing domains are taken from the regulatory domain of an H_2O_2 sensing bacterial protein (OxyR-RD), which contains an oxidizable cysteine residue. In the presence of reactive oxygen species, the two cysteine residues from the two OxyR-RD sensing domains become oxidized and create a disulfide bridge. The disulfide bridge induces a conformational change in the β-barrel and results in the fluorescence of the cpmApple. Thus, HyPerRed can be used as a sensor to detect the presence of reactive oxygen species that result from JNK-dependent PRDX6 recruitment and G protein pathway activation.

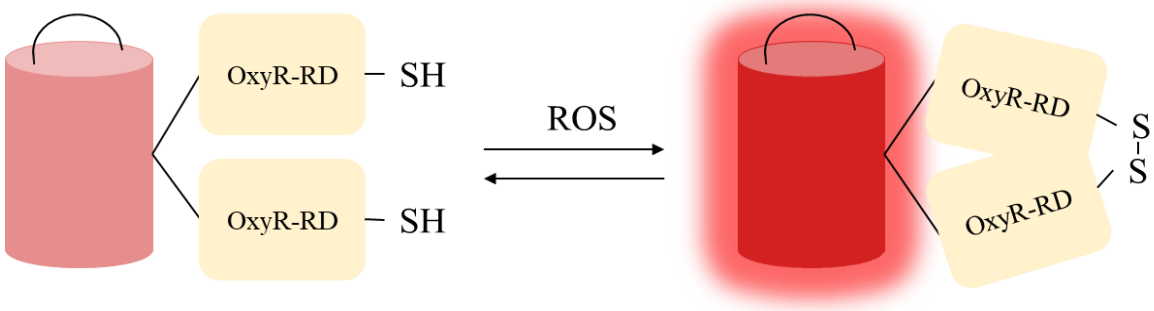


Figure 7: HyPerRed is a genetically encoded redox sensor that detects the presence of reactive oxygen species. *This sensor operates by using two cysteine residues on the OxyR-RD domain. (Left) In the absence of reactive oxygen species, the cysteine residues are reduced and do not form a disulfide bridge. In the presence of reactive oxygen species, these residues are oxidized and form a disulfide bridge. This results in a conformational change of the β-barrel which reduces the amount of quenching of the fluorophore due to water, and thus reports the presence of reactive oxygen species.*

A variant of the HyPerRed sensor, HRM63, was developed by the Berndt lab at the University of Washington. Instead of emitting red fluorescence, this variant was designed to emit green fluorescence in the presence of reactive oxygen species. Instead of using cpmApple as the fluorescent protein, HRM63 uses GFP as the fluorophore. In addition, HRM63 contains further modifications to the cysteine bridges to optimize fluorescence.

2.8.4 Genetically Encoded Neuromodulator Sensors

Another useful piece of information that is useful for more directly measuring neural circuit dynamics are sensors that fluoresce under receptor-binding dependent conditions. Patriarchi et al (2018) developed a genetically encoded dopamine indicator, dLight1, that fluoresces when bound to dopamine. This sensor is a variant of various dopamine receptors (dopamine D1 receptor, D2 receptor, and D4 receptor), in which the third intracellular loop is replaced with the circularly permuted fluorescent protein β-barrel from GCaMP6.

After creating the dLight sensor, they developed other intensity-based sensors based on different GPCRs, one of which being KOR. The indicator, kLight1.3, was developed in a similar manner, in which the cpGFP was inserted in the third intracellular loop of KOR.

This sensor is advantageous for directly measuring KOR activation by ligands such as dynorphin and can provide useful information about the direct effect of certain pharmacological agents. However, since it is a new sensor, it can still be optimized further to provide improved fluorescence levels.

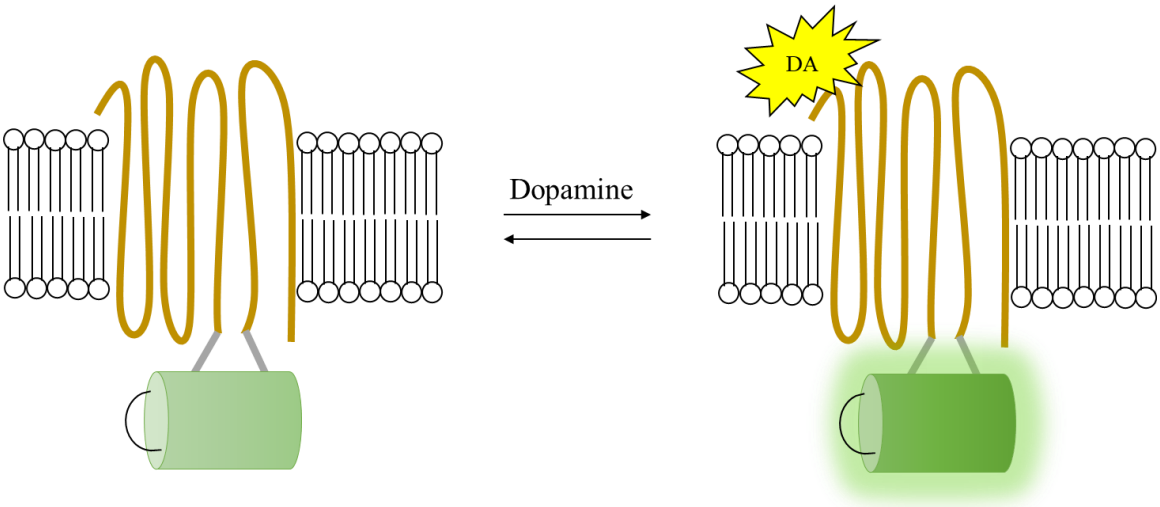


Figure 8: The dLight sensor serves as an indicator for the presence of receptor-specific ligands. *The dLight sensor was created by inserting a cpGFP into the third intracellular loop of the dopamine receptor. Binding of an agonist for the receptor, such as dopamine, results in a conformational change in the GFP, thereby resulting in the fluorescence. The variant used in this thesis was the kLight sensor, in which KOR is used in place of the dopamine receptor.*

2.9 Fiber Photometry

Fiber photometry has been a useful technique for analyzing fluorescence signals due to chemical species sensors for the last few years. Single fiber photometry is based on microscopy principles that allow for freely moving and behaving mice while performing a behavioral task.

Light is first emitted by a light source, typically an LED with a wavelength around the excitation spectrum of sensor being used. In the case with GCaMP, typical excitation wavelengths are blue shifted at about 473 nm (Gunaydin et al., 2014; Lerner et al., 2015; Wright et al., 2017). The light is passed through an excitation filter that isolates wavelengths in the sensor's excitation spectrum and is reflected with a dichroic mirror. It is then passed through an optical fiber that is implanted over the brain region of interest. In the presence of the chemical species of interest, the light will be absorbed by the sensor and will emit a new wavelength of light in the sensor's emission spectrum. In the case with GCaMP, typical emission wavelengths are green shifted at around 512 nm. The emitted light is transmitted through the optical cable and through the dichroic mirror into an emission filter that filters out wavelengths outside of the emission spectrum. The light is passed to a photodetector which records the bulk fluorescence level in the brain region of interest.

A variation of this fiber photometry system utilizes two light sources at different wavelengths. One wavelength acts as the excitatory wavelength for a sensor such as GCaMP, while the other wavelength acts as a control, or isosbestic, channel. By using both channels, it allows for correction for motion artifacts. Real signal will be present only in the GCaMP channel, whereas any fluorescence change due to motion artifact will appear in both channels and can be subtracted off from the GCaMP channel.

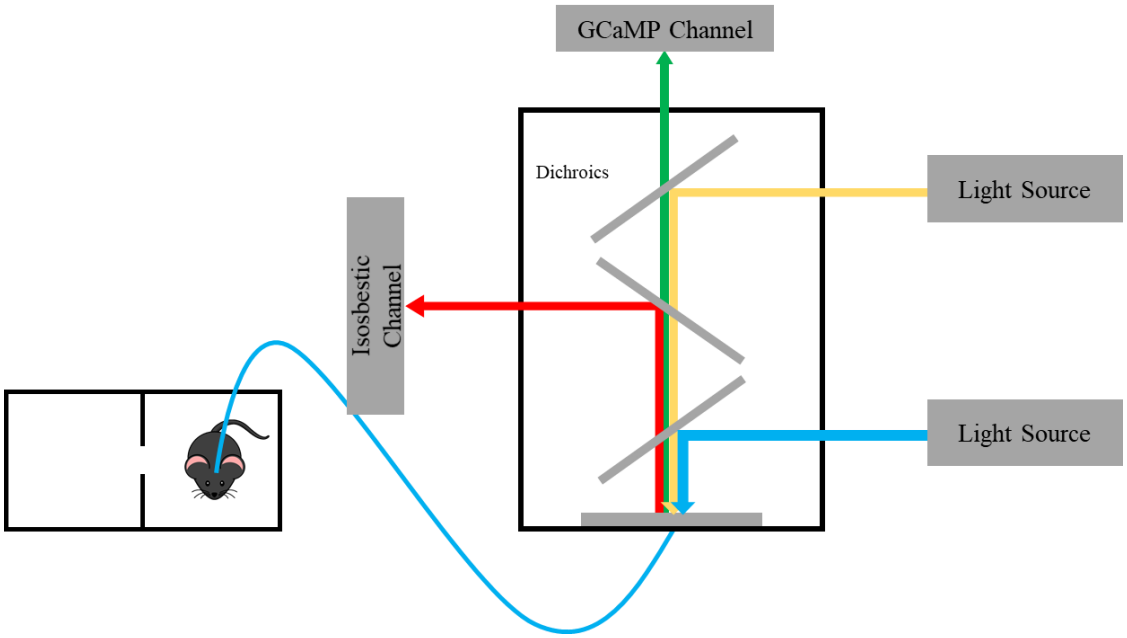


Figure 9: Fiber photometry measures the bulk level fluorescence of a population of neurons expressing genetically encoded sensors. *In the isosbestic fiber photometry setup, two light sources are reflected off dichroic mirrors and passed through a fiber optic cable implanted in the mouse skull over the brain region of interest. Fluorescence from the sensors are then emitted back through the fiber optic cable. The light is then reflected off one dichroic specific for the isosbestic wavelength into the isosbestic channel, and the other light is passed through another dichroic specific for the green channel.*

2.10 Effects of KOR on Behavior

After investigating the effects of different pharmacological agents on KOR at a cellular and neural circuitry level, we then turned to investigating how KOR influences changes in behavior. Disruption in KOR function have been shown to lead to anxiety and depression-like symptoms in animals (Carlezon et al., 2005) and stress-like effects (Land et al., 2008), while KOR antagonists have been shown to counteract some of these effects (Land et al., 2008; McLaughlin et al., 2003; Mague et al., 2003). Other studies illustrate the importance of KOR in modulating behavioral responses by selectively deleting KOR from specific neuronal subpopulations using Cre recombinase and observing a reduction of conditioned place aversion in those animals (Ehrich et al., 2015). These experiments show that KOR function has important implications on behavioral phenotypes that can be manipulated and investigated as a potential target for treating substance use disorder. Some of the main behavioral effects involve motivation for drug seeking behaviors and effects on cognition

and working memory. We investigated both facets through various behavioral assays, which will be discussed in the following sections.

2.10.1 KOR and Drug Reward: Conditioned Place Preference

To model the motivation for drug seeking behaviors under stressed conditions, we used a cocaine conditioned place preference paradigm. Briefly, conditioned place preference experiments are performed by first allowing the mice to freely explore a two-compartment chamber, where each chamber is distinct from the other. The amount of time spent in each chamber is measured and used as a baseline. The mice then undergo conditioning periods, in which they are injected with the drug of interest or saline and restricted to one side of the chamber. During the post conditioning test period, they are allowed to freely explore both compartments of the chamber and the amount of time spent in each chamber is recorded. The conditioned place preference score is measured by subtracting the amount of time spent in the drug-paired chamber in the post test from the amount of time spent in the drug-paired chamber in the pre-test. These experiments illustrate the drug seeking behavior and motivation for drug reward in mice. By disrupting KOR function and performing these experiments, it provides insight as to the role of KOR in mediating these types of behavioral responses. A variant of this conditioned place preference is conditioned place aversion, in which a similar paradigm is used, and the main difference is that an aversive stimulus is delivered instead of drug.

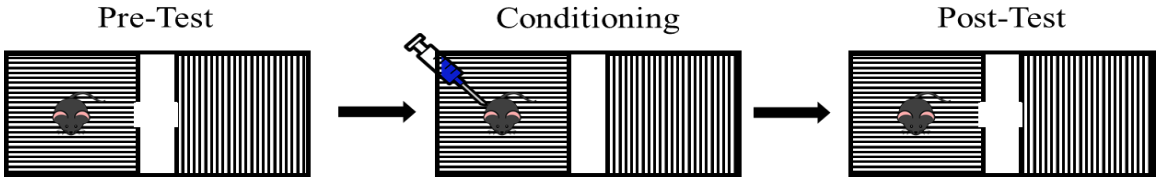


Figure 10: The general behavioral paradigm for conditioned place preference experiments. *Mice are initially allowed to freely explore both compartments of the conditioning chamber, with distinct cues in each compartment, and the time spent in each chamber is measured. During the conditioning days, the mice are restricted to one chamber and are administered the drug of interest, and the mice thereby associate the distinct compartment with the drug. After the conditioning days, the mice are once again allowed to freely explore both compartments. The time spent in the drug paired chamber is measured and the pre-conditioning test time is subtracted from the post conditioning test time to yield the preference score.*

Previous studies have investigated the effects of forced swim stresses on drug potentiation by using the cocaine conditioned place preference paradigm, which suggests that chronic stressors could activate the KOR-dynorphin circuit and thereby result in an increase in stress-induced potentiation of cocaine (McLaughlin et al., 2003). The role of the KOR-dynorphin system was further investigated in later studies involving the effect of timing of KOR activation prior to stressful events on cocaine conditioned place preference (McLaughlin et al., 2005; Chartoff et al., 2015). These findings suggest that the timing of KOR activation prior to stressful events plays an important role in contributing to either suppression or potentiation of stress-induced cocaine conditioned place preference. Further studies investigate how KOR contributes to the stress-induced dysphoric response, implicating p38α MAPK in VTA dopaminergic neurons as a contributor for conditioned place

aversion (Ehrich et al., 2015). Taken together, these studies suggest that the timing and method of KOR activation can have variable effects on the potentiating effects of cocaine through a host of signaling cascades.

2.10.2 KOR and Cognition: Delayed Alternation

Besides affecting motivation and an increasing drive for drug seeking, KOR is also thought to mediate cognitive effects and working memory. The delayed alternation task used in this thesis is a behavioral assay that is an important indicator of such cognitive function in mice. Specifically, it investigates the functionality of the working memory in mice, which is distinct from long-term or short-term memory. Short-term memory refers to the ability to retain limited pieces of information in a temporary and very accessible state, while long-term memory refers to the storage of knowledge and recollection of prior events (Cowan, 2009). Working memory is similar to short-term memory but is still somewhat distinct. Working memory is regarded as a memory that is used to inform and carry out a specific behavior. Thus, the purpose of the delayed alternation tasks in this thesis is to assess the working memory in mice after a predetermined delay period in response to different pharmacological treatments and stressors.

Briefly, delayed alternation tasks are performed by placing the animal in a chamber with two levers. The animal must learn that, after an initial lever press on either lever, they will be rewarded if they correctly press the alternate lever. If they do not, they receive no reward. To test the animal's working memory, both levers retract after the initial press and reextend after a predetermined delay period. Experiments consist of a certain amount of training days, in which the animal learns the behavior. Once they have adequately learned the behavior, they then undergo a pharmacological treatment, surgical treatment, or some other kind of stressor, and the effects on working memory of those treatments are assessed by performing the delayed alternation task once again. Typical data readout is conducted as percent correct and total number of alternations.

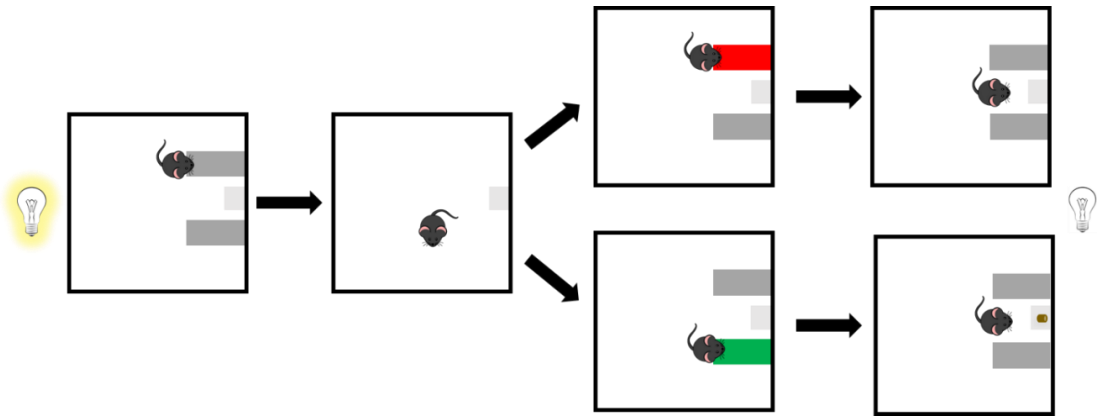


Figure 11: The delayed alternation task assesses cognitive disruption and working memory in mice after various treatments. *Mice are initially presented with two levers to press. Once the mouse presses one of the levers, both levers retract for a predetermined delay period. After the delay period, the levers extend once again. If the mouse incorrectly presses the same lever, the levers retract, and no reward pellet is delivered. If the mouse correctly alternates and presses the opposite lever, the levers retract, and a reward pellet is delivered. This test is repeated for 60 minutes.*

Alternation tasks in rats were performed initially to study the effect of lesions in the caudate nucleus on working memory, in which rats were rewarded after correctly alternating between lever presses (Chorover and Gross, 1963). However, there was no delay period between lever presses. This study served as the foundation for developing methods for testing working memory in rodent models and further iterations on this method introduced predetermined delay periods between lever presses.

Another study was performed to assess the effect of a tyrotrophin-releasing hormone analogue in rats with AMPA-induced lesions (Ballard et al., 1996). In this study, both an alternation task as well as a delayed alternation task was performed to train rats on the task and subsequently assess their working memory. Further experiments were later performed to investigate the effects of lesions in the primary and secondary motor cortices on working memory in rats (Yin et al., 2009), the effects of lesions, pharmacological treatments, and optogenetic stimulation in the medial prefrontal cortex (mPFC) on working memory in mice (Rossi et al., 2012), and the effects of chemogenetic inhibition of D1 receptor-positive (D1R+) cells in the lateral cerebellar nucleus (LCN) on working memory in mice (Locke et al., 2018).

Therefore, delayed alternation tasks have been well-established as effective methods for assessing the effects of various treatments and stressors on the capability to retain working memory in mice. As a result, it is an appropriate behavioral assay for assessing the effects on cognition resulting from disruptions in the KOR-dynorphin circuit.

3 Methods

3.1 Section Overview

This section provides the basis for the methods used in these experiments. It focuses on the cellular resolution, neural circuitry, and behavioral aspects of the experiments that were performed, as well as the motivation behind performing the experiments, the experimental methods, the data processing procedures, and the data analysis techniques.

3.2 Cellular

To better understand the dynamics of certain sensors, we performed various experiments at the cellular level to observe sensor fluorescence as a response to different conditions and pharmacological agents. These experiments provide information at a single cell resolution that allows for a better understanding for how pharmacological agents of interest act at a cellular level. The experiments are important for verifying and validating the use of such sensors described previously in animal models in these cellular level experiments.

3.2.1 *Ex vivo* Single Cell Imaging

Motivation
Cells expressing the genetically encoded sensor of interest were imaged in *ex vivo* slices to validate their functionality and measure the response of certain neuronal cell populations to pharmacological agents such as nalfurafine and naloxone.

While some information is gained from whole field imaging procedures, we looked specifically at individual cell fluorescence as well as the fluorescence of the entire image. Since cells firing out of phase would result in a certain amount of signal attenuation, single cell resolution can provide a

richer set of information regarding the timing of certain cell types to the drugs of interest. These trials provided a simple method for measuring a sensor's capacity to accurately report the level of the species of interest. In this case, we specifically wanted to test the HyPerRed sensor and its ability to detect reactive oxygen species.

Experimental methods

To test the HyPerRed sensor *ex vivo* at a single-cell resolution, DAT-Cre mice were injected with an AAV-DIO-HyPerRed in the VTA to express in dopaminergic neurons. All experimental procedures were performed as described in Reichard et al (in revision). Mice were rapidly decapitated after 10-12 weeks, after which the brains were then sliced and incubated in a specialized oxygenated solution of NMDG. The slices were then perfused with aCSF and imaged at 555 nm on an upright fluorescence microscope. After a 14-minute baseline period, the slices then had nalfurafine perfused over them to induce KOR activation mediated by G protein signaling pathways. A separate group of slices were pretreated with naloxone, after which they had nalfurafine perfused over them to observe the effect of an opioid receptor antagonist on G protein biased agonism with nalfurafine. All *ex vivo* experimental procedures and image recordings were performed by Katie Reichard and Keionna Newton in the Chavkin Lab (Reichard et al., in revision).

Image Configuration

Stacks of image were acquired in a .tif format. These images were loaded into ImageJ to verify broad trends in cellular fluorescence. They were then saved in a .jpg format by saving as an image sequence to be handled by MATLAB. Once they were converted to jpg, they were then run through the custom written MATLAB (version R2015a) script, whose code is included in **Appendix II**. The path to the folder containing the script was defined, as well as the folder containing the image data. The images were then read into MATLAB using the `imread()` function and sorted to ensure that the images were in the proper order. Once the images were read into MATLAB, they were saved into a .mat variable format to be used in the main script.

Image Processing

Once the images were in a MATLAB variable format, they were read into the image analysis script, which is included in **Appendix II**. The folder path that contained both the .mat variable file and the script was defined. The frames were read in and saved in a data structure to be analyzed later. The frame in which the drug treatment was delivered was defined as well as the frames to be used as the baseline period. Each individual baseline frame was analyzed using the `imfindcircles()` function in MATLAB to find all circles with a radius between 4 to 15 pixels and with a sensitivity of 0.9. For each of the circle centers that were identified, the corresponding neighborhood indices were recorded using a morphological disk-shaped structuring element using the `strel()` and `getnhood()` functions. The structuring element was converted to a vector of linear indices using the `find()` function, which was then converted to x- and y-coordinates about the center of each identified circle using the `ind2sub()` function.

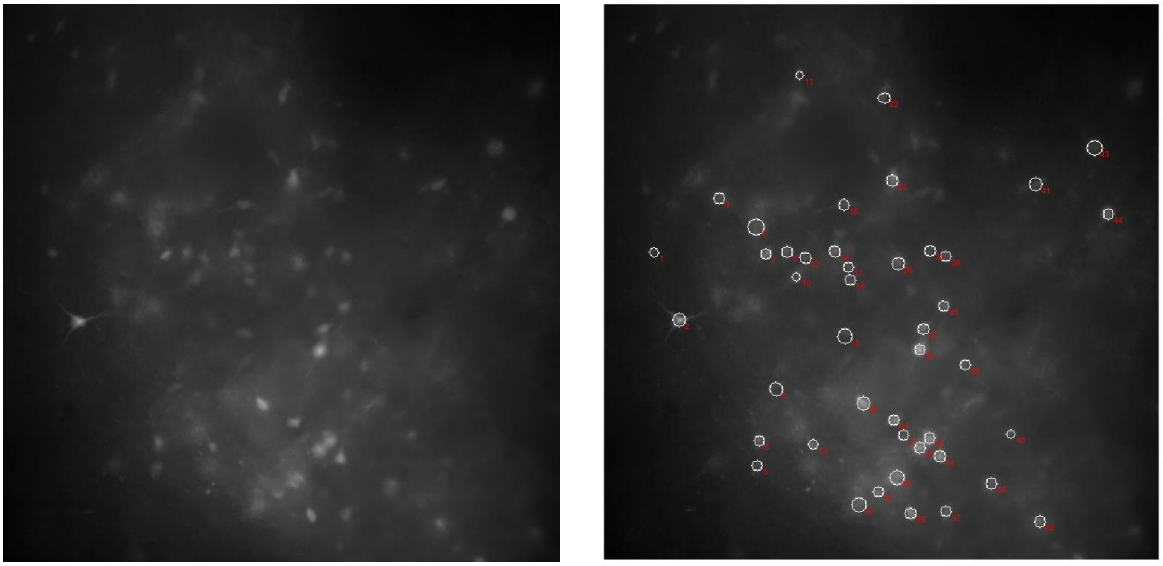


Figure 12: Cells were defined using a custom written cell detection algorithm. *(Left) Original characteristic frame from ex vivo single slice imaging. Images were taken of HyPerRed expressing dopaminergic neurons in the VTA. (Right) Original frame with cell masks detected by the automated algorithm overlaid. The red numbers correspond with the identification for each cell throughout the course of the recording.*

Once all of the circles had corresponding neighborhood x- and y-indices, they were added to the x- and y-coordinates for the centers of each circle and the cell indices were binarized to yield a mask for all of the identified regions of interest. A separate structure for the borders of each of the baseline cells was created by using the `bwperim()` function on each of the filled cells. The corresponding baseline cell signal was recorded by summing the values for each of the cell indices in the original frame.

Image Analysis

To create a baseline signal value for the recording, the binarized cell regions during the baseline frames were overlaid to create a binarized composite baseline frame. A structure and pixel index list for connected components was created using the `bwconncomp()` function. The centroid and area for each connected component was extracted using the `regionprops()` function. The baseline signal was then created by summing the pixel values for each region of interest and normalizing by that region's area for each of the baseline frames. The baseline frames were then averaged to generate an individual baseline fluorescence value for each of the regions of interest.

Fluorescence values for each region of interest for each frame were then generated in a similar manner as the baseline fluorescence values by summing each pixel value and normalizing by the region of interest area. They were then normalized to $\Delta F / F_0$, where F_0 was the baseline fluorescence value for each region of interest.

The processes used for image processing and image analysis are summarized in **Figure 13.**

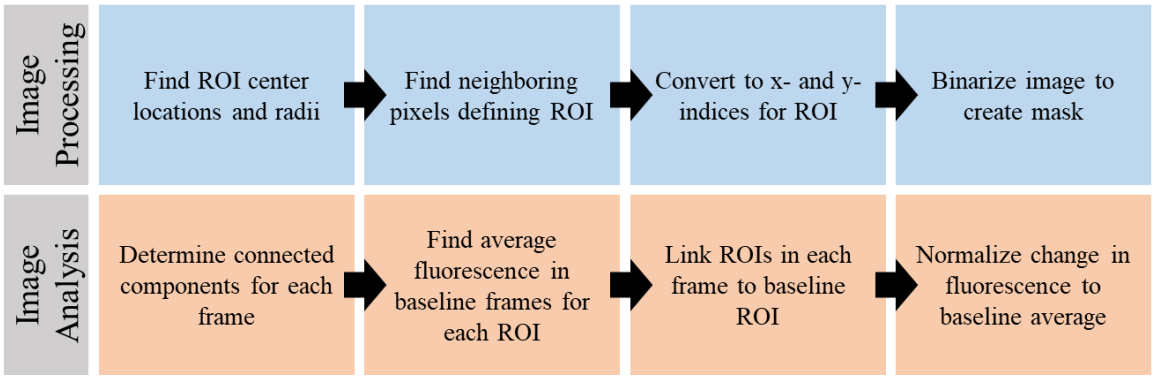


Figure 13: Schematic illustrating the general procedure for processing and analyzing ex vivo single cell images. *Images were processed by identifying center locations and radii for each ROI in the frame using imfindcircle(). A structuring element for the corresponding size of circle was created to determine surrounding pixels that define the ROI using strel() and getnhood() functions. The elements were then converted to x- and y-indices by using find() and ind2sub() functions, which were then added to the center for each ROI to generate a solid mask for the cells. Borders of each ROI were created by using bwperim(). Images were analyzed by first determining the connected components for each frame and averaging the fluorescence for baseline frames for each ROI. The ROIs were then linked to track individual cells over the course of the recording and were then normalized to reflect ΔF / F.*

3.2.2 Single HEK Cell Imaging

Motivation

Single HEK cell imaging is a useful tool for initial phases of developing and validating the use of certain protein sensors for the detection of specific chemical species. While the physiological composition of HEK cells does not necessarily accurately represent that of neuronal cell types, it can still generate useful insights as to how those protein sensors interact with the chemical species of interest generally. It is a cost-effective method to run these initial tests without having to deal with complications implicated with the use of animal models and can provide supporting evidence and motivation for using the sensors in those animal models for future studies.

Experimental Methods

HEK293 cells expressing mycKOR were transfected with genetically encoded sensors using preestablished methods (Schattauer et al., 2019). These sensors included HyPerRed, Hy46 (a precursor to HRM63), and HRM63. The cells were imaged using real-time cellular microscopy to monitor the fluorescence of both sets of cells. Recording sessions consisted of around 80 frames, with a 60 second interval between each frame recording. To test functionality of the sensors, initial tests were performed by using an H_2O_2 ramp and observing increases in fluorescence from the sensors. In subsequent trials, the pharmacological agent of interest was washed on between frames 10 and 11 and an H_2O_2 ramp was washed on at the end of the recording to confirm capability of detecting reactive oxygen species.

Initial testing consisted of expressing mycKOR infected HEK cells with Hy46 to verify functionality of the reporting domain of the sensor. An H_2O_2 ramp was used at various increasing concentrations on half of the positions on a plate and a single H_2O_2 wash on the other half of the

positions to observe the ability of the sensor to report levels of reactive oxygen species. This was replicated twice, with a different live imaging media used on the second plate. After these validation trials were performed and further optimizations were made on the sensor, the variant HRM63 was then tested in the HEK cells with H_2O_2 on half of the positions on the plate and vehicle on the other half to observe the ability of this sensor to report levels of reactive oxygen species. Finally, HRM63 was tested with KOR agonists on half of the positions and vehicle on the other half to observe the ability of the sensor to report levels of reactive oxygen species in a physiologically relevant environment.

By using U50,488 or nalfurafine as the pharmacological agents of interest, it allowed visualization of the reactive oxygen species production of individual cells. This was done by measuring the fluorescence values of the first ten frames as the baseline and normalizing the fluorescence in each of the frames to those baseline values. Additionally, the whole field fluorescence was observed to compare with individual cell fluorescence.

All HEK cell experimental procedures and imaging were performed by Selena Schattauer in the Chavkin Lab. HyPerRed variants such as Hy46 and HRM63 were provided by the Berndt Lab.

Image Configuration

Each set of images was composed of two plates with four positions, each with a recording channel for HyPerRd and HRM63. Initial images were generated in a .lif format, which were then loaded into ImageJ to perform preliminary filtering steps to aid in the cell detection algorithm. The images were bandpass filtered to filter small structures up to 3 pixels and filter large structures down to 40 pixels. Once the images were filtered, they were saved as .gif files to be run through a custom written MATLAB script for extracting frames for each of the recordings.

The .gif files were stored in a data folder and were loaded into MATLAB using the `dir()` function. Once the files were stored, the images were read using the `imread()` function. They were sorted into structures for containing stacks of images over the course of each recording for both HyPerRed and HRM63 channels. The stacks of HyPerRed and HRM63 images were then saved as a MATLAB variable to be loaded into the cell tracking script.

Image Processing

The schematic in **Figure 14** shows the workflow for how the stacks of images were processed and analyzed.

The .mat variable files for both the filtered and original set of images were loaded into the cell tracking script and the specific plate, position, and sensor type were defined. The frames to be used as baseline frames were also defined. For each of the baseline frames, the cell identification algorithm was run.

A subtraction mask was created for each frame by thresholding it to the mean value of the frame and filtered with at 2D Gaussian smoothing kernel by using the `imgaussfilt()` function. The subtraction mask was then subtracted off of the frame, which was then filtered again.

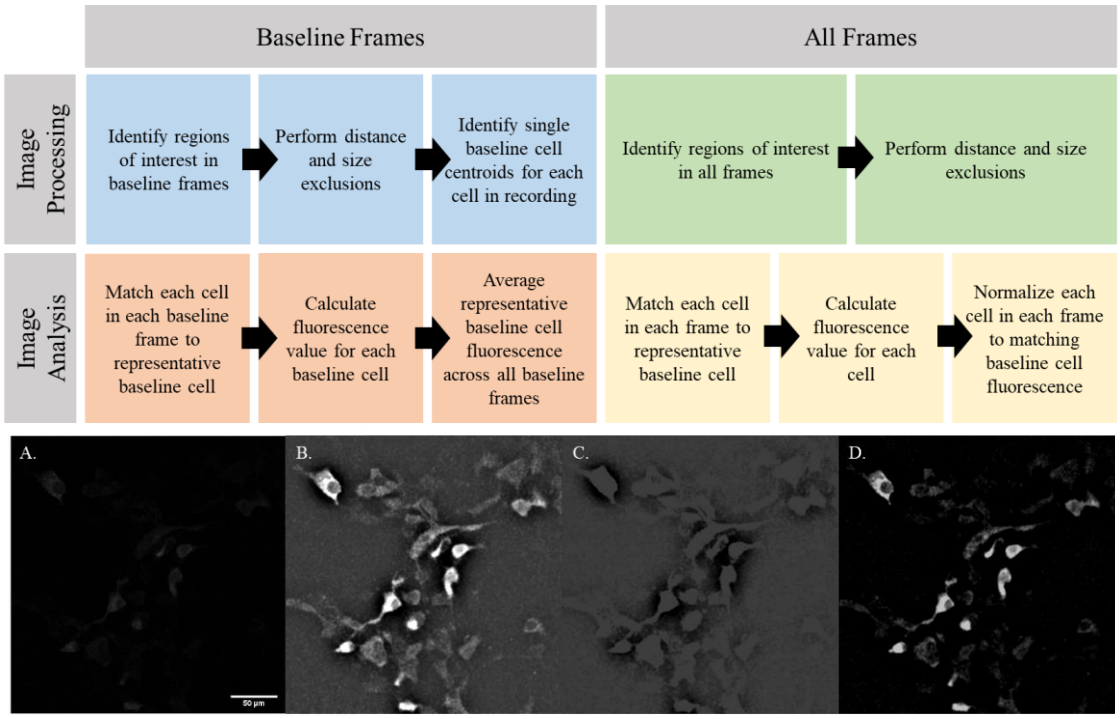


Figure 14: Schematic illustrating how the frames for each experiment were processed and then analyzed. *(Top) Specific methods were used to identify regions of interest in the baseline frames and identify characteristic cells to track throughout the rest of the recording. The rest of the frames were processed in a similar manner to generate cell masks. The cells throughout the recordings were matched to the identified baseline cells to allow for cell tracking. (Bottom) (A) The original frame was converted from a .lif format to a stack of images that were read into MATLAB. These images were used primarily for quantifying the actual fluorescence levels of the images. (B) The bandpass filtered frame was generated in ImageJ for ease of cell detection and was used primarily for creating the cell masks. (C) The bandpass filtered frames were processed by using a subtraction mask algorithm: a threshold was applied to the frame and filtered using a 2D Gaussian smoothing kernel, which was subsequently subtracted from the original bandpass filtered frame and was filtered again. (D) The resulting processed image highlights the cells and eliminates background noise to enable accurate cell detection and masking.*

The border for each region of interest was identified by using an edge-detection algorithm previously developed (Andrews, 2019). Borders were generated by parsing through the frame pixel by pixel and identifying neighboring poin ts. Frame values were stored for the center pixel as well as the frame values for pixels above, below, left, and right of the center pixel. For each pixel, the upper and lower pixels are compared with set threshold values. If one is greater than the threshold value and the other is less than the threshold value, the pixel is determined to be a border pixel. Similarly, the left and right pixels are compared, and if one is greater than the threshold value while the other is less than the threshold value, it is also determined to be a border pixel. The borders matrix was then filtered by using a morphological closing function of dilation followed by erosion and ensured that all features overlapping with the edges of the frames were closed at those overlapping points. Once the borders were fully defined and processed, the borders were filled to yield binary labeled images for the regions of interest in each frame. Small regions were filtered out by using the `bwlabel()` and `bwareaopen()` functions. The `bwdist()` function was then used on the complement of the filled binary images to compute the distance transform, or the distance of each pixel to the nearest non-zero pixel, which was then negated and run through the

`imextendedmin()`, `imimposemin()`, and `watershed()` functions. This allowed for the segmentation of cells that were in contact.

After individual cells were identified, a structure for connected components was created using the `bwconncomp()` function and the number of cells along with the pixel index list was extracted from the structure. A size exclusion was used to exclude any objects that had an area less than a certain threshold of pixels. The centroids, areas, and pixel index list for each of the remaining cells was then stored.

Figure 15: The general workflow for identifying regions of interest in each frame. *(A) Borders for each of the regions of interest are generated using an edge-detection algorithm. The cells had borders defined based on neighboring pixels for each pixel of the frame. (B) Cells were filled, and small regions were filtered out. (C) A watershed algorithm was used to aid in cell segmentation. This allowed for the detection of cells that were in contact. (D) Cells were finally converted back into borders for a mask for all the regions of interest. This algorithm was applied for all frames in the recording.*

A distance exclusion was then performed on the list of baseline centroids in the first baseline frame by using the `ismembertol()` function with a tolerance of 0.03. All centroids that failed this exclusion were averaged together to create a composite centroid and composite pixel index list. Following this initial exclusion, all centroids in each baseline frame were compared with the list of current baseline centroids. Any centroid in the baseline frames that failed an exclusion with a tolerance of 0.05 was considered a unique cell and therefore added to the list of baseline centroids.

For processing all frames of the recording, a similar procedure to baseline region of interest identification was used to identify regions of interest for each recording frame. Regions with an area less than 400 pixels and that had less than 40 pixels between each other were excluded.

Image Analysis

After the regions of interest were identified and processed, the cells were matched to their respective baseline centroid to track single cells over the course of the recording. The distance of each centroid in each frame to each baseline centroid was calculated and the minimum distance indicated the corresponding baseline cell. This was repeated for every frame and was organized into a matrix with each row corresponding to a distinct baseline cell and each column corresponding to each frame of the recording.

Once the corresponding baseline cell for each region of interest in the baseline frames was identified, the regions were overlaid across all baseline frames for each representative baseline cell to create a composite pixel index list. This composite pixel index list was then used as a mask on the original baseline frames to sum the fluorescence values in those indices and yield a single baseline fluorescence value for each representative baseline cell for each baseline frame. The fluorescence

for each representative baseline cell was then averaged across all baseline frames. Cells that were not detected in at least two of the baseline frames were excluded.

All frames were then processed using a similar method to match each region of interest to a representative baseline cell. For each region of interest, a mask of the corresponding pixel index list was created and stored to the representative baseline cell number for each frame. These masks were then used to sum the fluorescence values for each region of interest in each original frame. After the raw fluorescence values for each frame was determined, they were then normalized to the corresponding representative baseline cell fluorescence.

3.3 Neural Circuitry

After verification of the functionality of the various sensors employed at the single cell resolution, we observed the correlations between fluorescence signals and behavioral output in freely awake and behaving mice. We sought first to develop and validate the use of various sensors *in vivo* for use in behavioral experiments. These experiments provided further insights into neural circuits and how specific brain regions connect and influence other brain regions as well as behavioral responses. Using these developed and verified sensors can help us to better understand neural connections and the effect that certain types of stressors can have on those connections. All experimental methods and fiber photometry recordings were performed by Antony Abraham and Sanne Casello in the Chavkin Lab.

3.3.1 HyPerRed Fiber Photometry

Motivation

While GCaMP tracks and reads out the presence of calcium as fluorescence, HyPerRed indicates the presence of reactive oxygen species. Sensing this chemical species can provide information about activation of the G protein, or arrestin-independent, pathway. Thus, it is useful to utilize biased KOR agonists while using HyPerRed to readout their action in a physiological setting. Validation of HyPerRed in *ex vivo* slices and in HEK cells provided the motivation to use this reactive oxygen species sensor in freely behaving mice to monitor bulk changes in reactive oxygen species concentration under various pharmacological treatments.

Experimental Methods

To test the sensitivity of the HyPerRed sensor, we injected mice with various pharmacological agents to see the response to reactive oxygen species generated from KOR activation. We injected mice with saline, nalfurafine, U50,488, naloxone and nalfurafine, and CMPD101 and U50,488. **Table 3** summarizes the injection scheme for each of the pharmacological treatments.

Table 3: Summary of pharmacological treatments for mice expressing HyPerRed sensor.

	Injection 1	Injection 2	Injection 3
Saline	10 ml/kg saline		
Nalfurafine	50 μg/kg nalfurafine		
U50,488	1 mg/kg U50,488	10 mg/kg U50,488	
Naloxone/Nalfurafine	10 mg/kg naloxone	50 μg/kg nalfurafine	
CMPD101/U50,488	15 mg/kg CMPD101	1 mg/kg U50,488	10 mg/kg U50,488

To read out bulk neural activity, we injected AAV-DIO-HyPerRed in the PFC. We collected the fluorescence data using the fiber photometry system previously described.

Data Configuration

Initial experiments were performed using HyPerRed as an indicator for reactive oxygen species generation. Data was collected using the Tucker-Davis Technology fiber photometry system and Synapse software. The fluorescence was recorded at a sampling frequency of 1017 Hz. The fluorescence data for both the green channel and red channel were stored in structures called "tanks". Using the MATLAB function `TDTBin2Mat()`, a custom written function by Tucker-Davis Technologies for use with their fiber photometry system, the green and red streams were extracted from the tanks and the raw data were stored in vectors using the "ProcessTanks_2" MATLAB script included in **Appendix II**. However, for the HyPerRed experiments, only the red channel was used. The vectors were saved as a .mat MATLAB variable to be used in the analysis program. Fluorescence data for all the treatment groups were extracted and processed in this manner.

Data Processing

After extraction from the tanks, the .mat files were read into the HyPerRed processing script. Each of the treatment group files were stored in their respective data folders and read into the script using the `dir()` function. The time stamps for each of the injections in each treatment group were defined and a cell was created to contain all the data for each of the treatment groups. Within each of the treatment group cells, separate cell structures were created to contain each session's raw data.

Each of the sessions were downsampled by a factor of 100 and normalized using the average of the five-minute baseline period in the beginning of the recording (F_0) using the following definition:

$$\frac{\Delta F}{F} = \frac{F - F_0}{F_0} * 100\%$$

The sessions were then smoothed using a moving average filter and detrended to remove any initial motion artifact (**Figure 16**).

For each injection, the minimum session length was determined and all other sessions for that injection were truncated to that length. The injection signals were then extracted and were fitted to subtract off any residual motion artifact or linear drift.

Data Analysis
The data for each treatment group were truncated to the shortest recording length. The signals were averaged across all mice for each treatment group and compared. The standard error was calculated for each treatment group and plotted along with the average signals. All statistical analyses for fiber photometry recordings were performed with a two-way ANOVA, where the two factors were a repeated measure of time and drug. For HyPerRed, Dunnett's post hoc was used at every time point to compare each treatment with nalfurafine.

3.3.2 HRM63 Fiber Photometry

Motivation
To increase the sensitivities of the experiments to the presence of reactive oxygen species, we used the HyPerRed variant, HRM63. We tested whether this would allow us to better visualize the activation of the KOR induced G protein signaling cascades. While the HyPerRed sensor emits red shifted light, the HRM63 sensor emits green shifted light. As a result, we were unable to use the sensor along with GCaMP to measure calcium transients, but we tested if this sensor could better visualize changes in reactive oxygen species concentration.

Experimental Methods
To test the sensitivity of the HRM63 sensor, we injected mice with a biased KOR agonist to see KOR activation resulting from those treatments. We injected mice with saline and nalfurafine. **Table 4** summarizes the injection scheme for each of the pharmacological treatments. For increased sensitivity to reactive oxygen species, we expected to see an increase in fluorescence after KOR activation, like that of HyPerRed.

Table 4: Summary of pharmacological treatments for mice expressing HRM63 sensor.

Treatment	Injection
Saline	10 ml/kg saline
Nalfurafine	50 µg/kg nalfurafine

To read out bulk neural activity, we injected AAV-DIO-HRM63 in the PFC. We collected the fluorescence data using the fiber photometry system previously described.

Data Configuration
Like the HyPerRed fiber photometry data processing, fluorescence recordings collected at 1017 Hz were stored in tanks. However, for the HRM63 fiber photometry, a system with an isosbestic channel was used instead. The fluorescence data for the green and isosbestic channels were stored in the tank and extracted using the `TDTbin2mat()` function and saved as a .mat file. Fluorescence data for all the treatment groups were extracted and processed in this manner.

Data Processing

The HRM63 signals were processed in a similar manner to the previous fiber photometry recordings. The HRM63 and isosbestic channels were downsampled by a factor of 100 and the baseline period was taken to be 5 minutes prior to the injection and were truncated to cut off any signal prior to the baseline period. The HRM63 and isosbestic channels were then normalized and the isosbestic channel was fitted to the HRM63 channel using a least-squares method and subtracted to adjust for motion artifact. A full description on how this least-squares regression was performed will be provided in the GCaMP fiber photometry section.

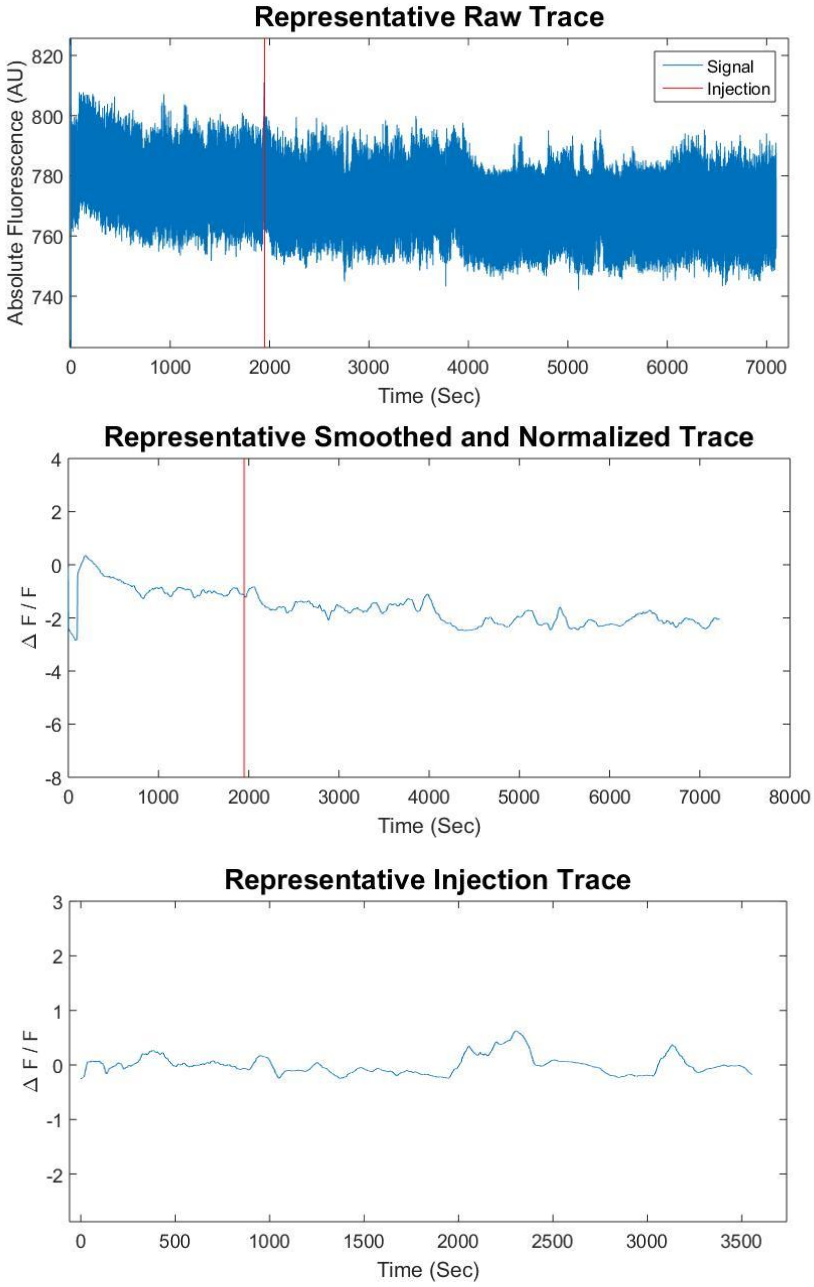


Figure 16: General processing for all fiber photometry data. *(Top) Raw data was initially loaded into MATLAB from tanks and saved as .mat variables to be used in other scripts. Raw data was sampled at a sampling frequency of 1017 Hz. (Middle) Data was then downsampled by a factor of 100, normalized to a set baseline period, and detrended to remove motion artifact. The signal was also low pass filtered using a moving average filter to eliminate high frequency noise. (Bottom) For each injection period, the signal of interest was extracted and truncated to the minimum recording length. These traces were again detrended to remove any residual motion artifact or drift to yield the final processed trace that was used for analysis.*

A new structure, `sortByTreatment`, was created to sort and contain the relevant experimental trials by treatment, which is summarized in **Figure 17**.

Each treatment group name was stored in the treatment field of `sortByTreatment`. The corresponding normalized data for each mouse was stored in cells in the `rawData` field. The injection times and mouse ID were stored in their respective field. A new field, `injections`, was created to store data relevant to each individual injection within each treatment. The data for each post injection period was stored in cells the `injectionData` field. The minimum length for each post injection period was determined and stored along with the corresponding minimum timeline for each injection period. After determining the minimum length for each post injection period, each post injection signal was truncated to contain the same number of data points and stored in a single array in the `stackedInjection` field.

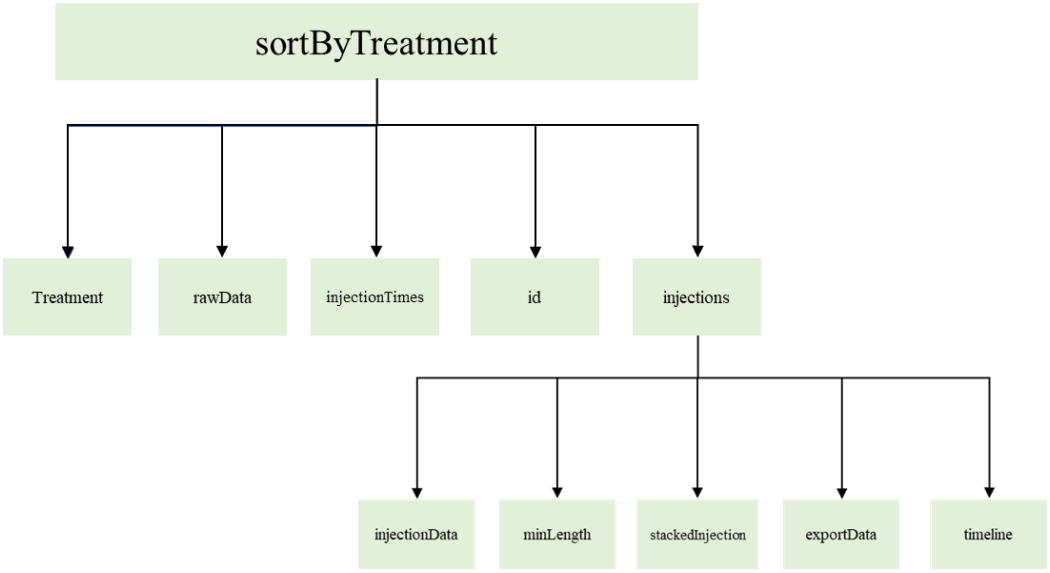


Figure 17: Flowchart showing the organization of the sortByTreatment data structure. *Initial fields for the treatment type, raw data, injection time stamps, unique mouse identification, and a nested structure containing information about the injections. The injections structure contains fields to hold the fluorescence data for each post injection period, the minimum post injection period length, all injections stacked in a matrix, data to be exported for analysis in other software, and a time axis for the post injection period.*

For ease of visualization, the data was further downsampled by a factor of 50, such that the new sampling frequency was 0.5 samples/sec. The stacked injection array for each post injection period and timeline were both downsampled by this factor and stored in a single matrix for export and ease of analysis.

Data Analysis

The data for each treatment group were truncated to the shortest recording length. The signals were averaged across all mice for each treatment group and compared. The standard error was calculated for each treatment group and plotted along with the average signals.

3.3.3 kLight Fiber Photometry

Motivation

While GCaMP, HyPerRed, and HRM63 sensors measure neuronal activity and KOR activation indirectly, the novel kLight sensor can measure KOR activation directly. This is beneficial for understanding the action of different pharmacological agents or stressors on the KOR-dynorphin circuit and can provide useful information that cannot be measured directly with the previously discussed sensors. Unlike HyPerRed and HRM63, which indicate which cells express KOR, kLight does not indicate KOR expression, but rather, indicates the action of ligands such as dynorphin that directly activate KOR.

Experimental Methods

To test the sensitivity of the kLight sensor, we injected mice with various pharmacological agents to see KOR activation resulting from those treatments. We injected mice with saline to record a baseline signal to compare with the other drug treatments. We also injected mice with 1, 5, and 10 mg/kg U50,488 to observe a graded fluorescent response to the unbiased KOR agonist. Additionally, we injected mice with 10 mg/kg U50,488 following injection with 10 mg/kg naloxone to see if the kLight sensor could report naloxone, an opioid receptor antagonist, blocking the action of U50,488 on KOR.

We also injected mice with 1 mg/kg naloxone after a repeated morphine dosing procedure to observe increases in dynorphin release from a stressor such as naloxone precipitated withdrawal. Finally, we injected mice with 50 and 150 μg/kg of nalfurafine, a G protein biased KOR agonist, to test if kLight could detect activation of the G protein pathway at various dosages. **Table 5** summarizes the injection scheme for each of the pharmacological treatments.

Table 5: Summary of pharmacological treatments administered to mice expressing kLight sensor. *Treatments were selected to investigate the kLight responses to an unbiased KOR agonist, stress evoked dynorphin release, and a biased KOR agonist.*

Treatment	Injection(s)
Saline	10 ml/kg saline
U50,488	1 mg/kg U50,488
	5 mg/kg U50, 488
	10 mg/kg U50, 488
	10 mg/kg naloxone, 10 mg/kg U50,488
Naloxone Induced Withdrawal	1 mg/kg naloxone
Nalfurafine	50 μg/kg nalfurafine
	150 μg/kg nalfurafine

To read out bulk neural activity, we injected AAV-DIO-kLight in the PFC. We collected the fluorescence data using the fiber photometry system previously described.

Data Configuration
Like the HRM63 fiber photometry data processing, fluorescence recordings collected at 1017 Hz were stored in tanks. The fluorescence data for the green and isosbestic channels were stored in the tank and extracted using the `TDTbin2mat()` function and saved as a .mat file using the "ProcessTanks_2" script. Fluorescence data for all the treatment groups were extracted and processed in this manner.

Data Processing
Data for the kLight fiber photometry was processed in a similar manner to the HRM63 fiber photometry processing, with the respective kLight treatment groups. The green and isosbestic channels were normalized with five minutes prior to the first injection taken as the baseline. After the normalization, both channels were low pass filtered and subsequently detrended to remove any linear drift. To account for motion artifact, the isosbestic channel was then fitted to the green channel using a least-squares regression method and the fitted isosbestic channel was subtracted from the green channel. The post injection signals were isolated from the motion artifact corrected signal for each mouse and treatment and stored in the `sortByTreatment` structure described previously. For each trace, the signal was vertically shifted to ensure each trace began at zero at the injection point and had comparable baseline values.

Data Analysis
In a similar manner to the HRM63 data analysis, the data for each kLight treatment group were truncated to the shortest recording length. The signals were averaged across all mice for each treatment group and compared. The standard error was calculated for each treatment group and plotted along with the average signals.

3.3.4 GCaMP Fiber Photometry

Motivation
While morphine withdrawal is one source of stress that could contribute to the pathways of addiction, we also investigated other sources of stress that also affect the addiction circuit in the brain. We used the existing and well-established sensor, GCaMP6m, to determine bulk neuronal activity in dopamine neurons in the VTA. These fiber photometry experiments that combined conditioned place preference and force swim stress provided a more complete picture of how repeated stressful events could potentiate cocaine preference in mice, and thereby be extended to human models.

Experimental Methods
To understand the effects that stress could have on mouse behavior, we performed a cocaine conditioned place preference experiment. In this paradigm, mice were tested and conditioned over a period of four days. Their response to the cocaine conditioning was measured through the difference of time spent in the cocaine-paired chamber after and before conditioning days.

On the first day, the mice were allowed to freely explore two chambers for 30 minutes, each with distinct characteristics (**Figure 18A**). The amount of time spent in each chamber was recorded and the chamber that the mouse spent the most time in was set as the drug paired chamber. The mouse was then placed in a bucket of water for 15 minutes to simulate a stressful event.

In the morning of the second day, the mouse was subjected a repeated forced swim stress test (rFSS), which consisted of four forced swim sessions, each lasting about six minutes (**Figure 18B and C**). Each force swim stress had a post period that also lasts around six minutes. After the rFSS test, the mouse was placed in the drug paired chamber and injected with cocaine. In this conditioning period, the mouse was not allowed to explore freely and was confined to the drug paired chamber for 30 minutes. In the afternoon on the second day, the mouse was placed in the opposite chamber and was injected with saline (**Figure 18C**). Again, in this conditioning period, the mouse was unable to explore freely and was confined to the non-drug paired chamber for 30 minutes.

On the third day, the mouse was conditioned in the same manner; they received cocaine injection in the morning in the drug paired chamber and a saline injection in the afternoon in the non-drug paired chamber (**Figure 18D**). By undergoing these conditioning phases, the mouse should associate the drug paired chamber with cocaine and should tend to spend more time in that chamber.

Finally, on the fourth day, they could freely explore both chambers for 30 minutes, and the time spent in each chamber was recorded (**Figure 18E**). The preference score was then recorded as the difference between the time spent in the drug paired chamber during the post-test and the time spent in the drug paired chamber during the pre-test.

In this specific experiment, three test groups were used; DAT mice with a wildtype expression of KOR in dopaminergic neurons, DAT flox KOR (DFK), which had a deletion of KOR in dopaminergic neurons, and an unstressed group with wildtype expression of KOR that did not undergo any of the force swim stress tests.

To read out bulk neural activity in dopaminergic neurons, AAV-DIO-GCaMP6m was injected in the VTA of DAT-Cre mice. This allowed for the visualization of activity of dopaminergic neurons in the VTA during forced swim stress tests as well as during cocaine conditioned place preference.

All conditioning and rFSS tests and fiber photometry recordings were performed by Antony Abraham in the Chavkin Lab.

Data Configuration

Like the HyPerRed fiber photometry data processing, fluorescence recordings collected at 1017 Hz were stored in tanks. The fluorescence data for the red and green channels were stored in the tank and extracted using the `TDTbin2mat()` function and saved as a .mat file. This methodology was used for all days of conditioning to yield separate .mat files for each mouse and for each day, as well as for each phase of the experiment.

Fluorescence data from days 1 and 4 were combined into a single MATLAB script to compare mouse reactions to cocaine before and after repeated force swim stress with subsequent conditioning periods. The portions of the fluorescence signal from day 2 with the repeated force swim stress tests were extracted and processed in a separate MATLAB script. Finally, cocaine and saline

conditioning periods in day 2 were combined with cocaine and saline conditioning periods in day 3 in a single MATLAB script for analyzing those conditioning periods.

Additionally, videos of the mice were recorded during each of the conditioning and repeated force swim stress sessions for behavioral analysis. These videos were analyzed using EthoVision by Noldus to track and determine behavioral characteristics of the mice. For example, data such as total distance moved, velocity, mobility state of the mouse, and zone location were determined and used to supplement the GCaMP fluorescence data.

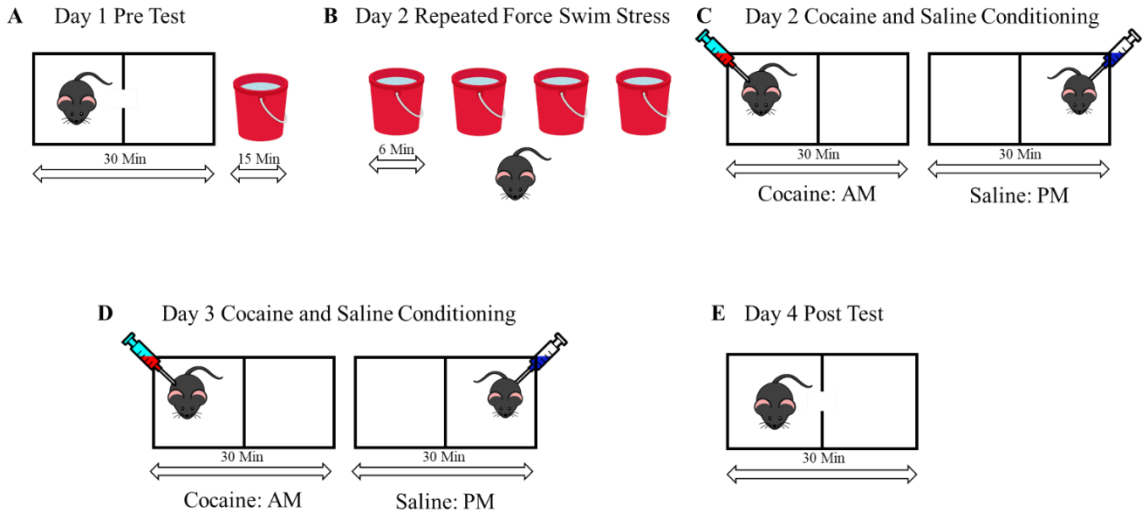


Figure 18: Paradigm used for cocaine conditioned place preference with repeated force swim stress tests. *(A) Mice were allowed 30 minutes to freely explore both compartments in the chamber and the time spent in each chamber was recorded. Immediately following the pre-test, mice were placed in a bucket of water for 15 minutes for a force swim stress. (B) On the second day, mice underwent 4 repeated force swim stresses, each lasting 6 minutes with a 6-minute post-swim period between each swim period. (C) Immediately following the repeated force swim stress test on the second day, mice were injected with cocaine and restricted to the drug paired chamber for 30 minutes. In the afternoon of the second day, mice were injected with saline and restricted to the non-drug paired chamber for 30 minutes. (D) The same procedure as in day 2 conditioning was performed on the third day. (E) Mice were allowed 30 minutes to freely explore both compartments in the chamber and the time spent in each chamber was recorded and compared with the pre-test times for the drug paired chamber.*

Data Processing: Days 1 and 4 Pre- and Post-Tests

For each of the analyzed periods, the GCaMP fluorescence signals were downsampled by a factor of 100. Since the recordings ranged anywhere from about 30 to 60 minutes with a sampling frequency of 1017 Hz, this resulted in several million points generated for each signal. This proved to be computationally restrictive for MATLAB to process, so the signals were downsampled by a factor of 100 to yield a new sampling frequency of 10 Hz.

For the day 1 pre- and day 4 post-conditioning periods, once the signals were downsampled, they were low pass filtered using a Butterworth filter with a frequency cutoff of 1 Hz to eliminate high frequency noise. To compare the fluorescence values between conditioning periods, both the GCaMP and red channel signals were normalized to $\Delta F/F$. The baseline fluorescence, F_0, was taken to be the mean value of the signal from 6 minutes to 1 minute before the mouse was placed in the conditioning chamber.

One challenge with fiber photometry is handling motion artifact. For example, due to the freely moving nature of the mouse and fiber photometry system, artifact can be introduced that is a response of movement and not a response of an increase in calcium levels. This can be handled by using an isosbestic control that is active at a different wavelength of light than GCaMP. In this experiment, we used a red channel to serve the same purpose. As a result, any change in fluorescence due to movement of the mouse will be recorded in both channels and can be removed by subtracting the red channel fluorescence from the GCaMP channel. However, since the amplitude of fluorescence is typically weaker for the isosbestic channel, it is often necessary to fit the isosbestic channel to the GCaMP channel, which can be done using a least-squares regression method.

Once the red and GCaMP channels were both normalized, the red signal was fitted to the GCaMP signal using least-squares regression. To do this, we used an iterative approach to determine the scalar multiplier that minimized the sum of squares between the red and GCaMP signals. Using the `fminsearch()` function, different scalar values were iteratively tested to find the best multiplier to fit the red signal to the GCaMP signal. The fitted red channel was then filtered and subtracted from the normalized GCaMP signal to yield a motion artifact corrected GCaMP signal (**Figure 19**).

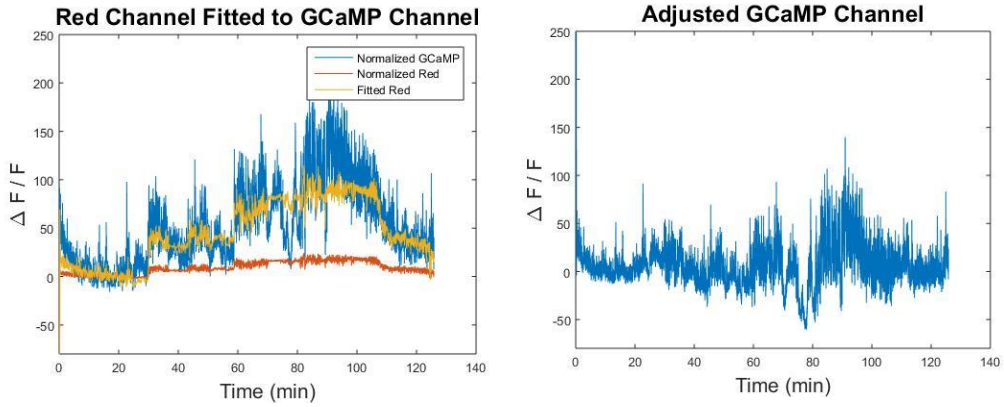


Figure 19: Removal of motion artifact from GCaMP fluorescence. *(Left) The original GCaMP signal was normalized to ΔF/F, but still retained some motion artifact. The red channel was also normalized to ΔF/F and was fitted to the GCaMP channel using a least-squares regression method. The fitted red channel was subtracted from the GCaMP channel to yield the motion artifact corrected signal (Right).*

After the GCaMP signal was motion artifact corrected, the 15-minute swim period for day 1 data was separated from the conditioning period.

The original downsampled data was additionally bandpass filtered from 0.2 – 1 Hz to highlight transients in calcium concentration. After the data was filtered, the pre- and post-conditioning periods were extracted and converted to a z-score for comparison across conditioning signals:

$$Z = \frac{F - mean(F)}{std(F)}$$

The `findpeaks()` function was used to identify transients in the filtered GCaMP signal. Using a minimum peak height of 2.9 standard deviations and a minimum distance between peaks of 10 samples, relevant transient locations were identified. Transients were further discriminated for by

excluding any transients that were both greater than 5 standard deviations and had a width of less than 2 samples, which were considered as artifact.

After loading in summary measures created from the EthoVision behavioral analysis, we determined when the mice were in the horizontal versus the vertical zone. The zone that the mouse spent the most time in during the pre-test on day 1 was designated as the drug-paired chamber, where the animal would receive cocaine injections on days 2 and 3. Based on that data, the drug-paired chamber for each mouse was determined and stored.

Data Analysis: Days 1 and 4 Pre- and Post-Tests

Combining the zone data from EthoVision and the GCaMP transient data for the pre- and post-tests, we were able to determine the number of transients that occurred in the paired and non-paired drug chambers for both the pre and post-test for each test group. This was accomplished by creating an array for when the mouse was detected in either zone. The starting zone was stored, and we looped over the entire length of the array. For each iteration of the loop, if the mouse was in the drug paired chamber and was not in the previous state, it was stored as a transition point to the drug paired chamber. If the mouse was in the non-drug paired chamber and was not in the previous state, it was stored as a transition point to the non-drug paired chamber. At the end of the iteration, the current location was stored as the last state for the next iteration to compare to.

Once the transition points had been determined, the GCaMP signal from -10 to 40 seconds on either side of the transition points for drug paired transitions and non-drug paired transitions for pre and post tests for each test group.

In addition to the transition points between chambers, we also determined the number of transients that occurred in drug paired chamber versus non-drug paired chamber for the pre and post tests for each test group. This was done by creating an array with the indices of both positive and negative transients. This array was looped over and if an index matched a time index of the mouse in the drug paired chamber, it was stored as a drug paired chamber transient. Conversely, if it matched a time index of the mouse in the non-drug paired chamber, it was stored as a non-drug paired chamber transient.

Data Processing: Day 2 Repeated Force Swim Stress Tests

The repeated force swim stress test GCaMP signals were processed in a similar manner to the days 1 and 4 signals. The GCaMP and red signals were downsampled, filtered, and normalized. The red signal was fitted to the GCaMP signal using a least-squares method and subtracted to adjust for motion artifact. The behavioral data was captured by using EthoVision to determine the amount of time mobile.

Data Analysis: Day 2 Repeated Force Swim Stress Tests

Once the GCaMP signal was processed, the signal was divided into 4 swim and post periods. The traces for each swim and post period were averaged across each of the test groups. Overall trends

in both the average swim and post periods were analyzed for each test group and the differences were compared.

Data Processing: Days 2 and 3 Cocaine and Saline Conditioning
The most important features for cocaine and saline conditioning in days 2 and 3 were the number of calcium transients that occurred while the mice were in the chamber. This was accomplished in a similar manner to the transient analysis in days 1 and 4 tests. The data was bandpass filtered, converted to a z-score, and had peaks identified by using the `findpeaks()` function. Positive, negative, and total transients were identified and recorded.

Data Analysis: Days 2 and 3 Cocaine and Saline Conditioning
The number of transients for each group was collected for both cocaine and saline conditioning on days 2 and 3. Various combinations of the groups were compared, such as comparing cocaine vs saline for each day or comparing each treatment across days. Additionally, each of these combinations were performed with positive, negative, and total transients.

3.4 Behavior

Using optical sensors in combination with pharmacological agents provides information about how those drugs affect signaling pathways and overall neural circuitry. Based on that knowledge, the effects of those pharmacological agents can then be further explored to see how those pathways influence behavior.

3.4.1 Delayed Alternation

Motivation
Previous studies have shown that KOR activation in VTA dopaminergic neurons is partially responsible for disrupting cognitive or stress-induced compulsive behaviors, and that KOR antagonists could be useful therapeutics for treating such behaviors (Abraham et al., 2018). By performing the delayed alternation task, we can use another method to visualize the effect that certain pharmacological agents or other stressors have on behavior and working memory in mice. We can investigate which brain regions and connections are important in the formation and regulation of working memory, and subsequently, how drugs of interest can affect those circuits and the resulting behavioral readouts.

Experimental Methods
For delayed alternation tasks, mice are placed in an operant chamber with the house light off. When the trial begins, the house light turns on and both right and left levers extend. The mouse can then

either press the right or left lever. Once the lever is pressed, both levers retract and a predetermined delay period of 2, 5, or 10 seconds begins. At the end of the delay period, the levers extend, and the mouse can press either lever. If the mouse incorrectly alternates and presses the same lever, no reward pellet is delivered. If the mouse correctly alternates and presses the opposite lever, a reward pellet is delivered. The intertrial interval of 20 seconds begins, the house light turns off, and the levers retract. Each session lasted 60 minutes.

The program connected with the operant boxes registers time stamps for every left and right lever press, reward delivery, when the house light turns on or off, when the right or left lever extends or retracts, when the intertrial interval begins or ends, head entry into the food tray, correct or incorrect alternation to the left or right, and when the delay starts or ends. These time stamps were stored and were saved to be analyzed using a custom MATLAB script to extract relevant information.

Mice were trained in the delayed alternation task until they sufficiently learned the task. Once they had learned the task, they were treated with their respective treatments, and the total number of responses and percent of correct responses were recorded. Four distinct groups of mice were run using the delayed alternation paradigm to investigate the effects of certain pharmacological agents, optogenetic stimulation, and withdrawal on cognitive function. Additionally, different stressors were tested to see which were sufficient to activate KOR induced signaling. All injections and treatments were done in the PFC **(**Abraham, Casello, Wong et al., in preparation**)**.

ACSF pretreated wildtype mice, norBNI pretreated wildtype mice, and mice with KOR in the PFC excised with Cre recombinase (PFC lox KOR) were injected with saline and evaluated using the delayed alternation task. They were then injected with U50,488 and their performance was evaluated using the delayed alternation task. Their performance was broken into distinct time bins and the percent correct and total responses for each bin were recorded.

For the second group, wildtype mice and PFC lox KOR mice were evaluated on the delayed alternation task with no treatment. They then underwent a repeated morphine dosing procedure and underwent naloxone precipitated withdrawal. During the withdrawal period, they were evaluated on the delayed alternation task. Additionally, mice expressing channelrhodopsin-2 (ChR2) in the PFC were evaluated on the delayed alternation task with no stimulation. They were then stimulated, and their performance was evaluated using the delayed alternation task.

Finally, control mice and PFC lox KOR mice were first injected with saline and their performance was evaluated using the delayed alternation task. They were then injected with nalfurafine and evaluated on the delayed alternation task. For each evaluation, the session was broken into 15-minute bins and the percent correct for each bin was recorded.

To understand which stressors were capable of effecting KOR induced signaling, mice were treated with various pharmacological agents and stressors. The percent change in KORp levels, an indicator for the β-arrestin pathway activation and phosphorylated KOR, relative to control were evaluated to determine which treatments were sufficient to result in KOR activation. These treatments were then pretreated with norBNI to investigate the extent of KOR dependence for the respective stressors.

All delayed alternation tasks, experimental procedures, and injections were performed by Antony Abraham in the Chavkin Lab.

Data Processing

To handle the large amounts of data produced by these delayed alternation experiments sampled at a rate of 100 Hz, we used a custom MATLAB script for extracting and storing all of the relevant information, based on preestablished code for the delayed alternation task (Andrews, 2019).

All files were relabeled to identify the date of the session and the chamber number, and each file had at most two boxes in which the mice were run. All relevant data files were stored in a single main data folder and read into MATLAB using the `dir()` function. Each file was divided into lines by using regular expressions and was split further into individual words. The number of total events, event identifiers, corresponding time stamps, and chamber number were recorded and stored in a structure designated for each file. The date of the file was determined by using a while loop to find the start date listed in the file. The date format was then coerced to be in the format "yy/mm/dd".

Dates on which each session occurred were identified by appending each date to a new structure and using the `unique()` function on that structure. A new structure, `sortByDate`, was created to organize each of the files into the dates in which each session was conducted. The number of correct and incorrect responses, percent correct, and total responses were stored in the `sortBy-Date` structure under each mouse's chamber and box number.

Once all the files had been sorted by date, chamber, and box number, the event and time matrices were analyzed to determine the timing of each lever press and alternation. The time stamps at which right and left alternations, correct right and left alternations, and incorrect right and left alternations occurred were saved in each structure within the `latencies` field. Once each of the time stamps for each kind of event was determined, the resulting lever presses were conveyed using a Raster plot to show the percentage of correct and incorrect alternations as well as the direction in which the alternations occurred (**Figure 20**).

Data Analysis

After extracting event time stamps from the recording files, we developed code to analyze session totals for percent correct and total alternations across all treatment groups. Additionally, we divided the entire session into different numbers of bins (2, 4, 6, and 12) and corresponding bin lengths (30 min, 15 min, 10 min, and 5 min) to analyze the number of correct responses, total responses, and percent correct in various phases of the session. This was performed by isolating the time stamps for correct and total alternations for each trial and performing a logic operation to include only stamps within the respective bin edge indices. The length of this vector was taken as the number of correct or total alternations within each bin, which could then be used to determine the percent of correct alternations. When comparing between groups in each experimental trial, a two-way ANOVA was performed with drug administration as one factor and group as another factor. Sidak's post hoc was then used to detect differences within each group.

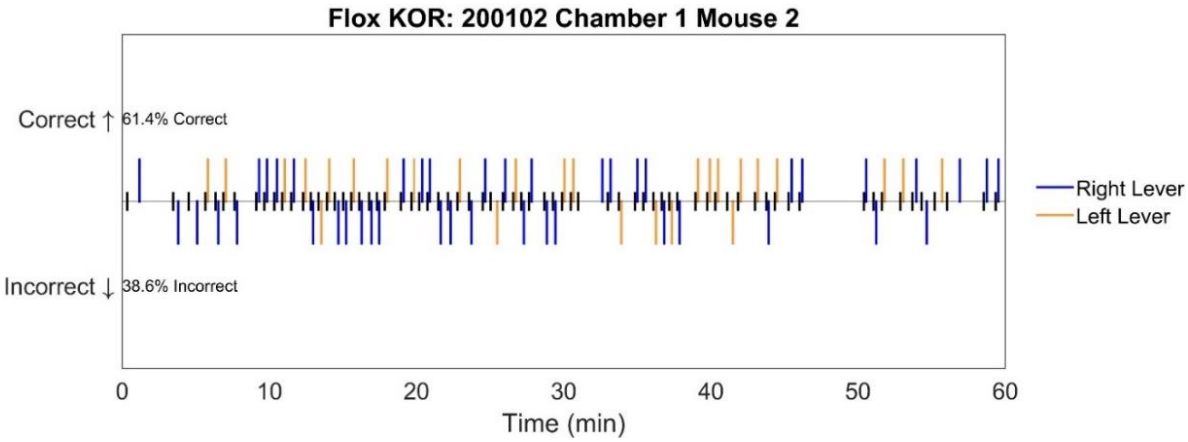


Figure 20: Representative Raster plot showing delayed alternation for a single mouse. *Relevant information regarding the number of correct and incorrect alternations were extracted from the files generated by the delayed alternation task and conveyed in Raster plots. These plots show the correct (upward lines) and incorrect (downward lines) alternations to the right (blue lines) or the left (orange lines).*

4 Results

4.1 Section Overview

The following sections communicate the results from each of the experimental procedures explained previously. The results have important implications that support the development and validation of various types of sensors. This section will focus primarily on the results of the computational methods and highlight key results of the experimental trials behind the computational analyses.

4.2 Cellular

Computational methods were developed to identify regions of interest in fluorescence microscopy images, match those regions of interest to cells, and quantify the cell fluorescence. These findings highlight the differences between population level fluorescence recordings and single-cell resolution fluorescence recordings. Additionally, they serve as initial tests for investigating the possibility of using the genetically encoded sensors in later *in vivo* systems.

4.2.1 *Ex vivo* Single Cell Imaging

For initial sensor testing and validation, DAT-Cre mice were injected with an AAV-DIO-HyPerRed in the VTA and allowed local sensor expression. After a period of 10-12 weeks, the mice were decapitated, after which the brains were sliced, incubated, and imaged. The nalfurafine group was generated by perfusing the slices with nalfurafine and imaging the resulting fluorescence. The other group was pretreated with the opioid receptor antagonist naloxone and then perfused with nalfurafine.

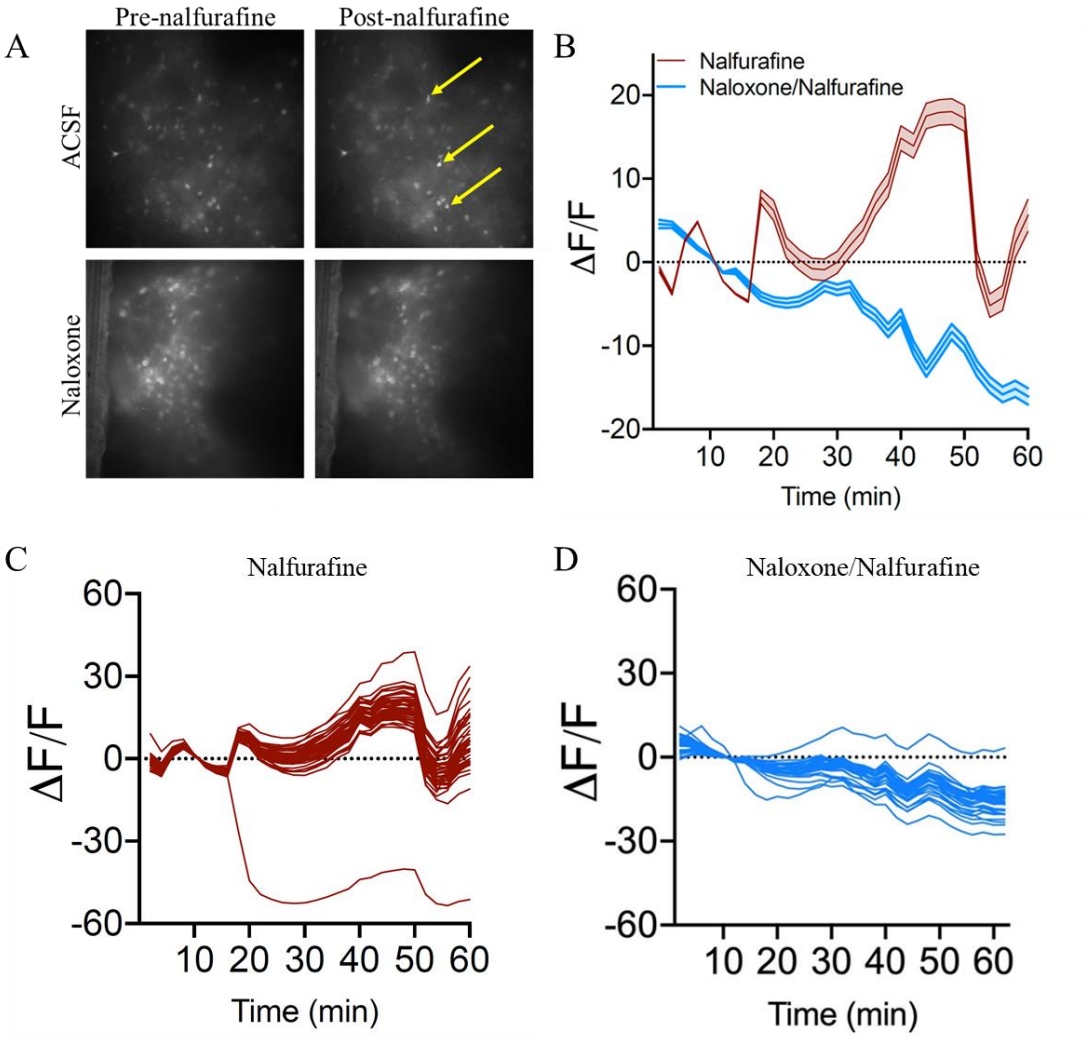


Figure 21: HyPerRed indicates the presence of reactive oxygen species in ex vivo single cell images. *(A) Representative images of VTA slices expressing HyPerRed before nalfurafine treatment (left column) and after nalfurafine treatment (right column). Pretreatment with naloxone (bottom row) inhibits fluorescence from treatment with nalfurafine compared with pretreatment with aCSF alone (top row). Yellow arrows indicate representative cells with obvious increases in fluorescence. (B) Average traces for individual cells expressing HyPerRed after nalfurafine treatment and naloxone/nalfurafine treatment. Nalfurafine shows an increase in fluorescence around 15 minutes post perfusion while nalfurafine treatment after naloxone pretreatment shows no increase in fluorescence. (C and D). Fluorescence traces for individual cells in either nalfurafine or naloxone/nalfurafine treatments.*

Without the presence of naloxone, slices that were perfused with nalfurafine exhibited an increase in fluorescence at around 15 minutes post perfusion. However, when the slices were pretreated with naloxone, this increase in fluorescence was not observed.

Additionally, there are slight variations from cell to cell in the amount of fluorescence. Individual cells react slightly differently to the treatment, illustrating the divergence between population level dynamics and single cell recordings. However, the general trend is still apparent, even with a population level analysis.

4.2.2 Single HEK Cell Imaging

HEK293 cells expressing mycKOR were transfected with Hy46 and HyPerRed to indicate the presence of reactive oxygen species as a response to increasing concentrations of H_2O_2. The response of those cells was recorded through fluorescence microscopy and processed using a custom MATLAB script. After identifying baseline fluorescence for each cell in the baseline frames and matching those cells throughout the rest of the recording, the cells were normalized to the respective baseline fluorescence. The computational analysis was able to detect fluorescence of individual cells and averaging the fluorescence of those cells per each frame closely matched values determined by visually inspecting the frames for regions of interest.

After validation with the initial HyPerRed variant (Hy46), the further optimized sensor (HRM63) was tested using similar techniques. HEK cells were treated with an H_2O_2 ramp to observe fluorescence in response to the presence of reactive oxygen species. **Figure 22A** compares the frame fluorescence for the Hy46, HyPerRed, and HRM63 sensors, respectively, at the beginning of the recording and after the 30 μM H_2O_2 wash. Comparing HRM63 fluorescence after the H_2O_2 wash with Hy46 and HyPerRed fluorescence after the H_2O_2 wash, the signal was weaker for HRM63 than Hy46 or HyPerRed.

After the H_2O_2 ramp, cells expressing HRM63 were then treated with either nalfurafine or U50,488 between the 10^{th} and 11^{th} frames, and were treated with H_2O_2 after the 50^{th} frame. Consistent with fluorescence based on regions of interest identified manually, there was little increase in fluorescence after the application of KOR agonists such as U50,488 or nalfurafine. While Hy46 and HyPerRed demonstrated increased fluorescence and increased reactive oxygen species sensitivity at physiologically relevant concentrations of reactive oxygen species, HRM63 did not demonstrate that same sensitivity.

4.3 Neural Circuitry

Through the development and validation of novel genetically encoded sensors, we were able to understand the dynamics that mediate KOR signaling pathways and effects of stress on the KOR-dynorphin circuit. We found that the different sensors provided different and unique insights into these neural circuits and the time course through which they were active.

4.3.1 Sensor Validation

As specific sensors provide different information regarding the signaling pathways activated through KOR agonists, we sought to develop and validate use of those sensors in a bulk population level setting by using fiber photometry. In comparison with GCaMP, which has been widely used in previous studies, these sensors still required validation for use in animal models. The sensors that we tested consist of the HyPerRed sensor, the HRM63 sensor, and the kLight sensor.

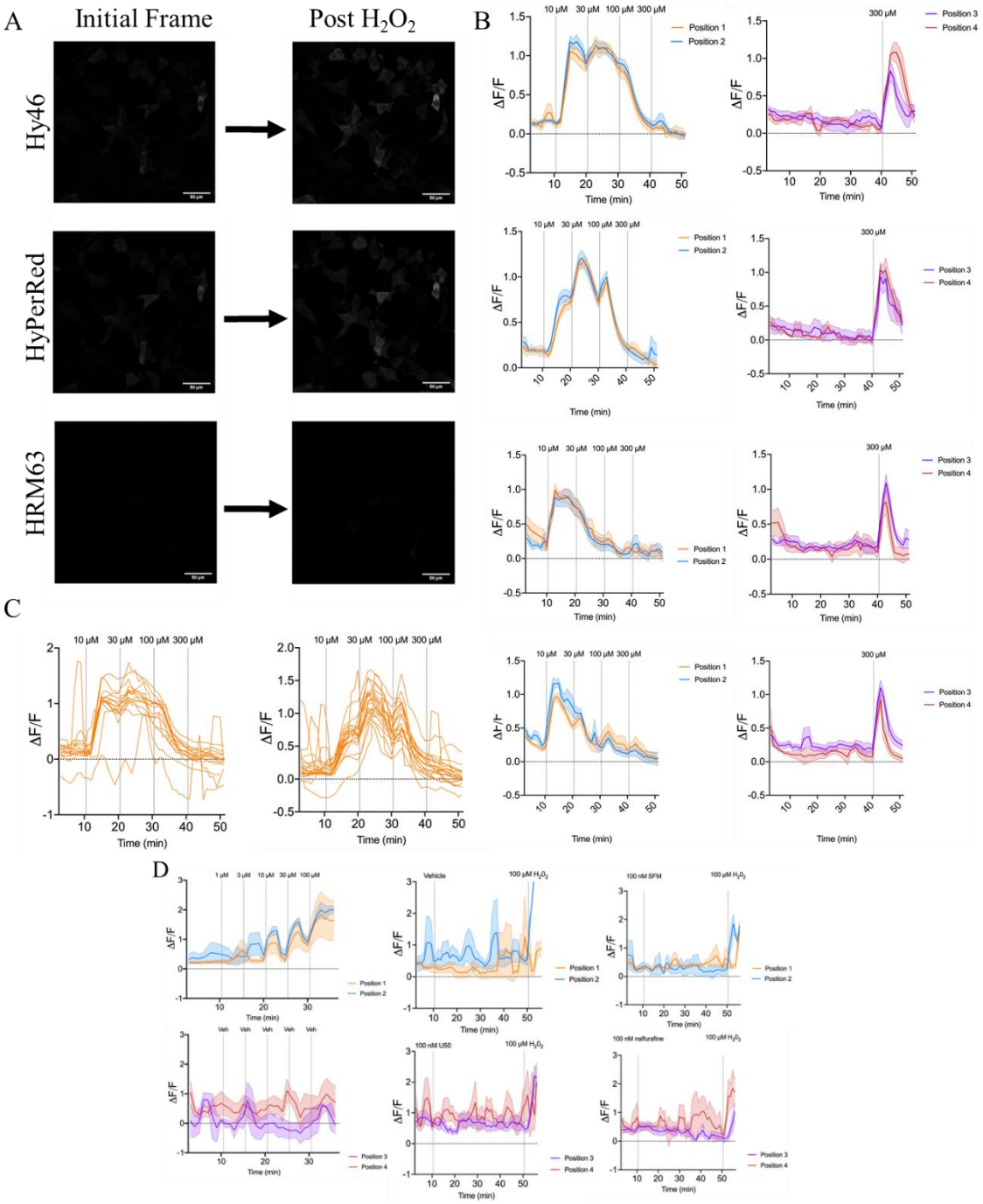


Figure 22: HEK cells transfected with Hy46, HyPerRed, and HRM63 were tested with an H_2O_2 ramp. *(A, Top) HEK cells expressing mycKOR transfected with Hy46 show visible increase in fluorescence between the 1st frame (left) and post 30 µM wash of H_2O_2 (right). (Middle) Similar to Hy46 trials, HEK cells expressing mycKOR transfected with HyPerRed show visible increase in fluorescence between the 1st frame and post 30 µM wash of H_2O_2. (Bottom) While not clearly visible, there is a smaller increase in fluorescence for HEK cells transfected with HRM63 after the 30 µM wash of H_2O_2 compared with that of either Hy46 or HyPerRed. (B) Average fluorescence for Hy46 (rows 1 and 3) and HyPerRed (rows 2 and 4) treated with an H_2O_2 ramp (left column) or a single wash of H_2O_2 at the end of the recording (right column). (C) Representative individual cell traces for the Hy46 H_2O_2 ramp for Hy46 (left) and HyPerRed (right). (D) Average traces of HRM63 treatments with an H_2O_2 ramp (left), U50,488 treatment (middle) and nalfurafine treatment (right).*

HyPerRed Sensor Validation

The HyPerRed genetically encoded redox indicator was used to investigate the time course of activation of the G protein pathway in response to a biased KOR agonist. Mice were injected with an AAV-DIO-HyPerRed virus in the PFC to induce HyPerRed expression in that brain region. The mice were treated with various pharmacological agents, most notably nalfurafine and naloxone, to observe reactive oxygen species generation resulting from the KOR mediated G protein pathway.

Nalfurafine induced an increase in reactive oxygen species concentration between 10 – 15 minutes prior to the injection time compared to mice injected with a saline control (**Figure 23A, B)**. Mice that were pretreated with naloxone and then injected with nalfurafine show inhibition of the increase in reactive oxygen species generation observed in the nalfurafine mice (**Figure 23C**).

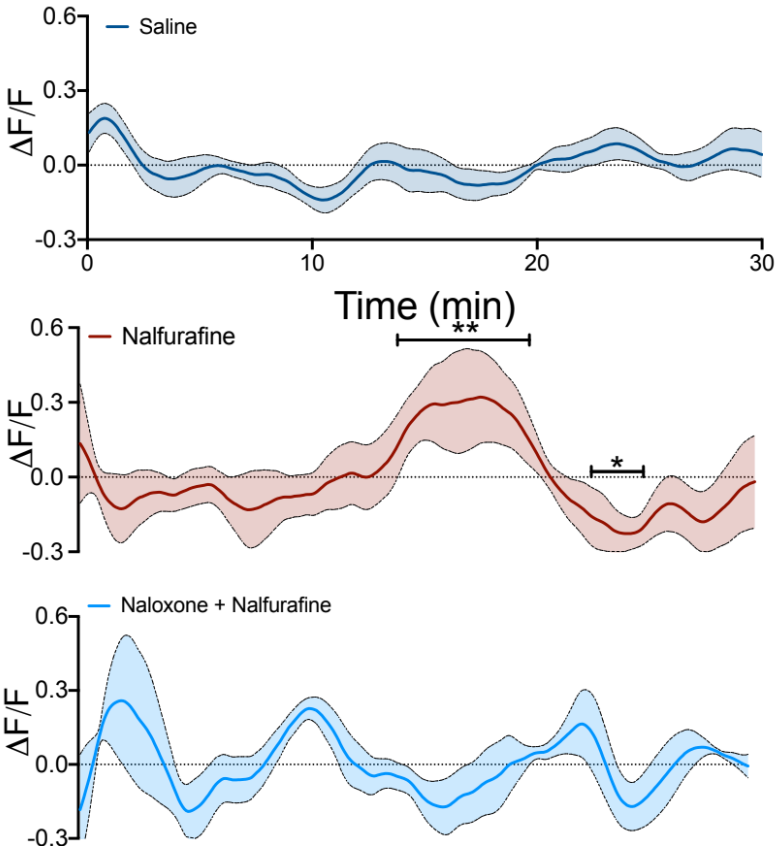


Figure 23: Validation of HyPerRed sensor using nalfurafine and naloxone. *(Top) Mice are injected with saline and the post injection period is shown. There is little change from baseline fluorescence levels. (Middle) Mice are injected with nalfurafine and the post injection period is shown. There is a peak in fluorescence that occurs about 15 minutes after the injection occurs. (Bottom) Mice are first injected with naloxone, an opioid receptor antagonist. They are then injected with nalfurafine, and the post nalfurafine injection period is shown. The increase in fluorescence observed in the nalfurafine injection alone is blocked when pretreated with naloxone (Abraham, Casello, Wong et al., in preparation).*

Findings from HyPerRed fiber photometry then prompted the use of HRM63 *in vivo* to investigate and compare the sensitivity between the two sensors. While a clear and statistically significant increase in HyPerRed fluorescence after nalfurafine treatment was observed, the sensitivity was lower than the sensitivity of other sensors such as GCaMP. Thus, we then tested HRM63 as a sensor in an *in vivo* system.

HRM63 Sensor Validation

Validation of the HRM63 sensor was performed by injecting mice with an AAV-DIO-HRM63 in the PFC to locally express the sensor. In a similar manner to other fiber photometry experiments, the fluorescence level was recorded after different pharmacological treatments and compared to injection of saline as the control. We found that after injection of nalfurafine, there was no appreciable increase in fluorescence compared with saline. Since we did not see any appreciable increase in fluorescence signal, we decided not to continue with further pharmacological treatments until expression of the sensor had been improved.

While both HyPerRed and HRM63 aim to investigate KOR activation and which cell populations express KOR, they did not provide information about the action of dynorphin and when it was released. Therefore, we required a different sensor that was not based on the presence of reactive oxygen species, but rather fluoresced under conditions in which KOR became activated by dynorphin. As a result, we then tested the kLight sensor *in vivo* to observe such effects.

kLight Sensor Validation

To test and validate the kLight sensor, we first injected mice with an AAV-DIO-kLight1.3 virus in the PFC to locally express the sensor. We then measured the fluorescence levels in the mice treated with various pharmacological agents including U50,488, nalfurafine, and naloxone using the fiber photometry system described previously. **Figure 24A** shows representative traces of mice after treatment with 10 mg/kg of U50,488 compared to after treatment with saline.

We found that after injection with U50,488, there was an increase in kLight fluorescence compared with saline that persisted for around one hour. By varying the dosages of U50,488, we observed varying degrees of kLight fluorescence. Additionally, we found that when pretreated with naloxone, increases in kLight fluorescence from U50,488 treatment was effectively blocked (**Figure 24B**). Treatment with naloxone after a repeated morphine dosing procedure induced an increase in kLight fluorescence similar to that of a mid-dose of U50,488 (**Figure 24C**). Finally, treatment with nalfurafine, a G protein biased KOR agonist, resulted in no effective increase in fluorescence compared with saline (**Figure 24D**).

After testing and validation of these genetically encoded sensors, this allowed us to determine which sensors would appropriately probe the scientific question of interest for further behavioral studies. Each sensor had specific advantages and limitations that influenced under which conditions they should be utilized, and as such, the correct selection of a sensor for a certain experimental procedure allowed for a new perspective into the dynamics being investigated.

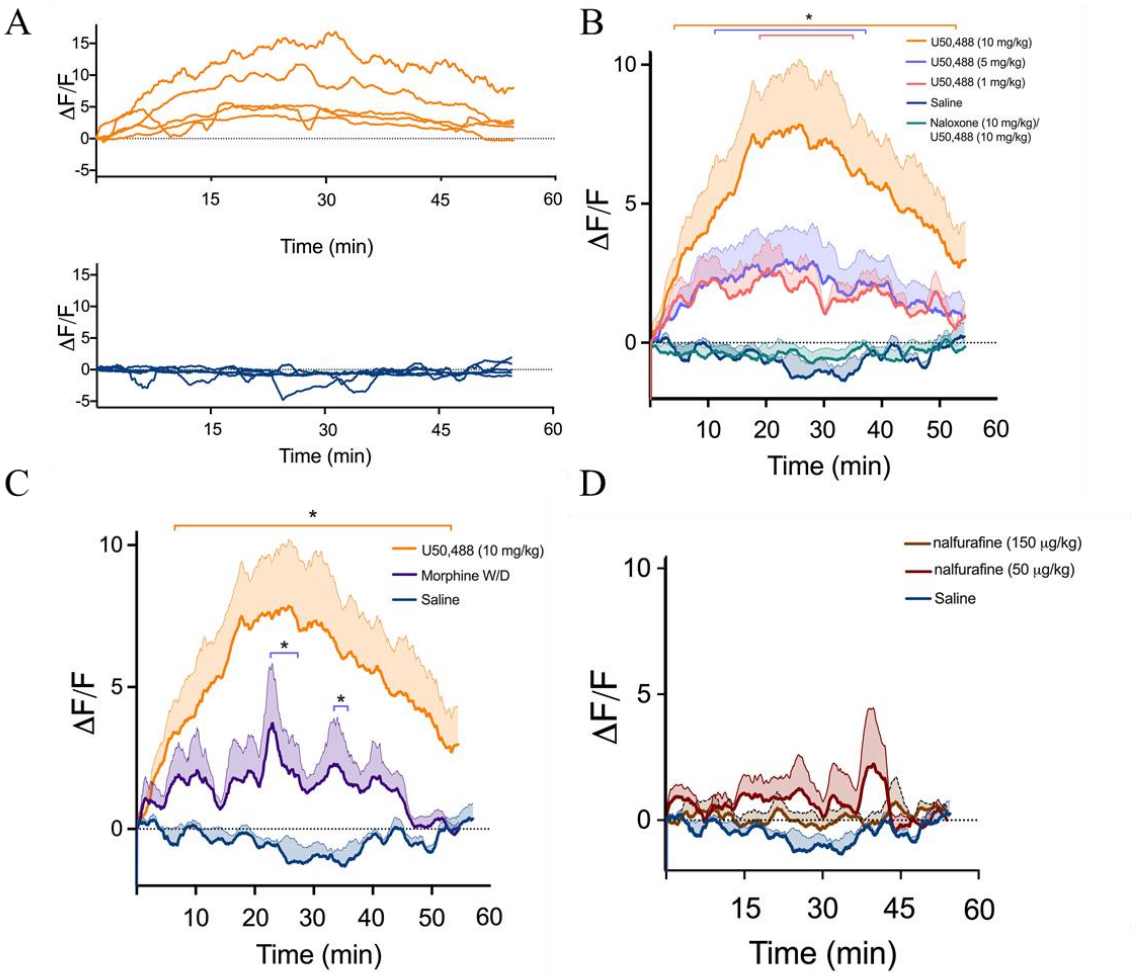


Figure 24: KLight effectively indicates full KOR agonism in a graded manner. *(A) Individual traces of kLight fluorescence for 10 mg/kg U50,488 (top) versus saline (bottom). (B) Increasing dosage of U50,488 results in an increased kLight fluorescence response compared to saline. Pretreatment with naloxone effectively blocks increases from KOR activation from U50,488 and is not significantly different from saline. (C) Comparison of kLight fluorescence after treatment with U50,488, naloxone precipitated withdrawal, and saline. Naloxone precipitated withdrawal results in an increase in kLight fluorescence, similar to levels of the 5 mg/kg dose of U50,388, but does not increase fluorescence to the same level as the 10 mg/kg U50,488 dose. (D) kLight fluorescence from treatment with either 50 or 150 μg/kg of nalfurafine is not significantly different from saline.*

4.3.2 GCaMP Fiber Photometry

After performing a cocaine conditioned place preference experiment, we were able to observe the effects of chronic stressors on addiction-like behaviors in mice. Mice were divided into three groups, consisting of wildtype and stressed mice, DAT flox KOR and stressed mice, and unstressed wildtype mice. After being injected with viral vectors containing GCaMP sensors and implanting a recording fiber over the VTA, mice were run through a conditioned place preference paradigm to observe changes in fluorescence and thus calcium activity in response to various conditions. These

recordings were handled and processed through custom written MATLAB scripts to normalize fluorescence values and extract relevant features of interest.

Days 1 and 4 Pre- and Post-Tests

As discussed previously, mice expressing GCaMP were placed in the conditioning chamber on days 1 and 4 to serve as the pre- and post-test, respectively, with fluorescence levels measured through fiber photometry. Preference scores were measured as the amount of time spent in the drug paired chamber during the pre-test subtracted from the amount of time spent in the drug paired chamber during the post-test. While this preference score gives useful information as to the potentiation of cocaine, we also utilized fiber photometry to highlight key aspects of the response at a neural circuit level. In the wildtype DAT mice, we observed little change in the fluorescence level when the mice transitioned to the drug paired chamber. However, during the post-test, there was a robust increase in fluorescence during transitions to the drug paired chamber. In mice lacking KOR in dopaminergic neurons, this increase during the post-test was lost, implicating KOR in contributing to the potentiating effects of cocaine (**Figure 25**).

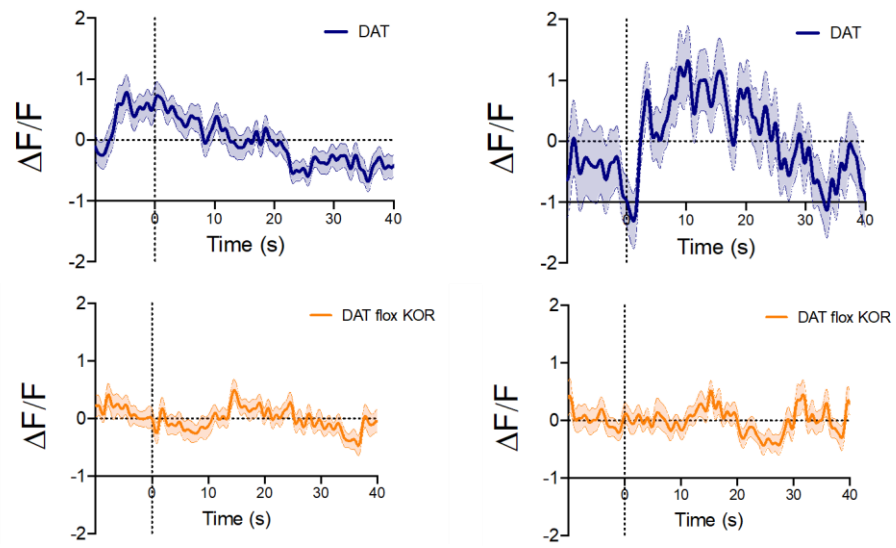


Figure 25: Dopaminergic VTA neuron activity during chamber transitions for pre- and post-tests. *(Top) Increases in fluorescence for wildtype mice during transitions to the drug-paired chamber in the post-test (right) were observed compared with the fluorescence levels for wildtype mice during transitions to the drug-paired chamber in the pre-test (left). (Bottom) No difference in fluorescence level was observed during transitions to the drug-paired chamber for DAT flox KOR mice in the pre-test compared with the post-test (Abraham et al., in preparation).*

Day 2 Repeated Force Swim Stress Test

Following the pre-test on day 1, DAT flox KOR and wildtype mice were subjected to a repeated force swim stress test while measuring fluorescence levels to observe activity of dopaminergic VTA neurons. The fluorescence level was measured during the swim and post-swim periods to compare the effect of chronic stressors for both groups of mice. Additionally, the average fluorescence was measured for swim and post-swim periods. In the wildtype mice with normal KOR expression, during the swim period, there was a depression in fluorescence during the beginning portion of the test. Following the swim test, there was an upward deflection in fluorescence.

Compared with the DAT flox KOR mice, which had a deletion of KOR in dopaminergic neurons in the VTA, there was an absence of either depression or increase in fluorescence for the swim or post-swim period, respectively (**Figure 26**).

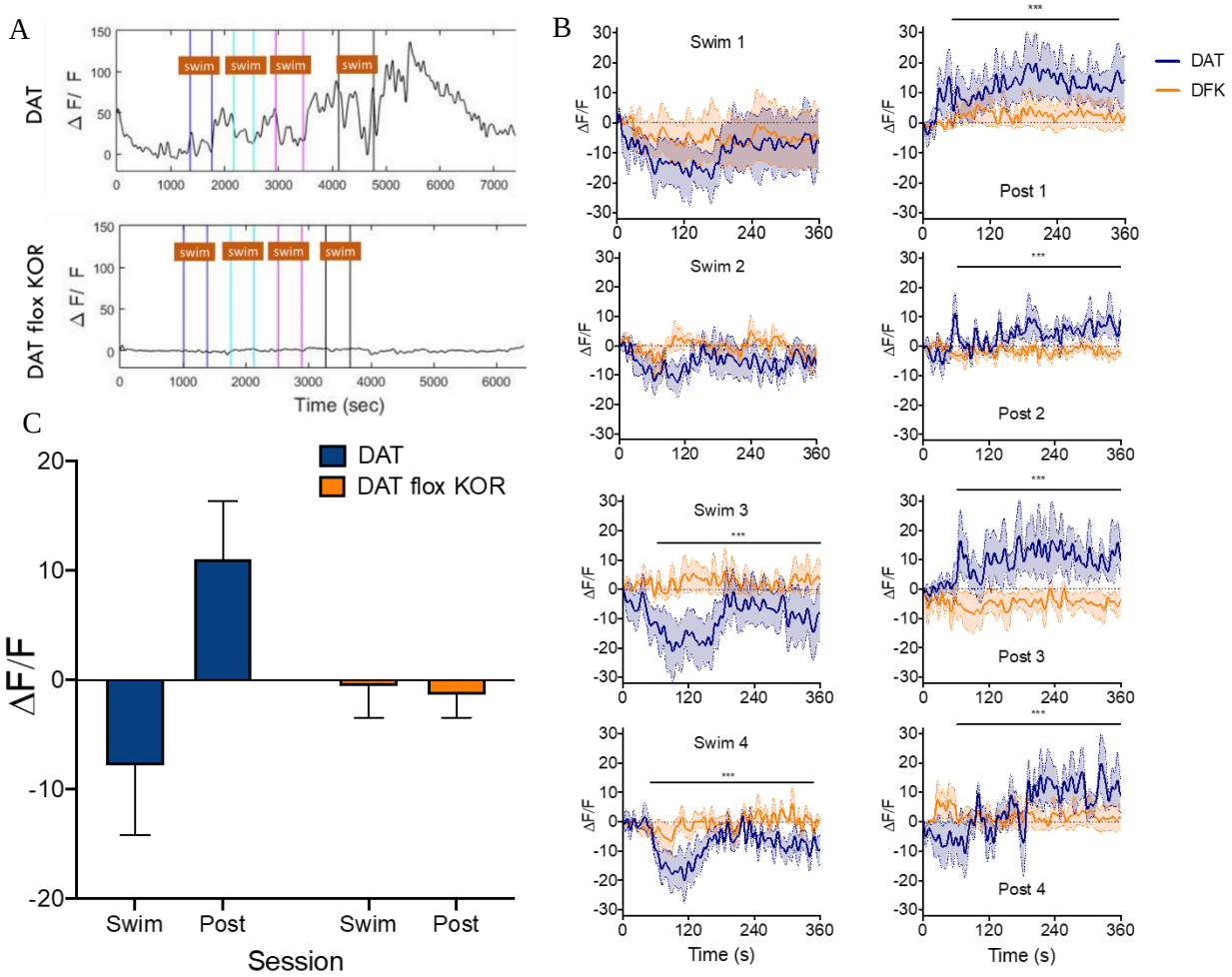


Figure 26: Repeated force swim stress induces decreases in fluorescence during swim periods and subsequent increase during post swim periods. *(A) Overall time course of fluorescence for representative wildtype and DAT flox KOR mice. Mice were subjected to forced swim stresses lasting 6 minutes with an equal length post period. (B) Averages of each individual swim and post period were extracted from the overall recording and overlaid for both wildtype and DAT flox KOR mice. During the first half of the swim stress, there is a characteristic decrease in fluorescence for the wildtype mice, which is absent in the DAT flox KOR mice. Subsequent post periods observe an increase in fluorescence for wildtype mice but not for DAT flox KOR mice. (C) Overall average fluorescence during swim and post periods for both wildtype and DAT flox KOR mice. Consistent with (B), the averages for wildtype mice during the swim period are negative with the average fluorescence for the post period being positive (Abraham et al., in preparation).*

4.4 Behavior

We found that at a phenotypic level, regulation of the KOR-dynorphin circuit controls behavioral outputs observed in animal models, and can be affected by different types of stressors, including pharmacological agents and various aversive contexts. These behavioral phenotypes can manifest as disruptions in motivation for drug seeking behaviors and cognition.

4.4.1 Delayed Alternation

While the cocaine conditioned place preference experiments provided information regarding stress induced KOR activation and motivation for drug reward, the delayed alternation experiments were performed to investigate the effects of KOR agonists and antagonists on cognition and working memory. This has important implications regarding these pharmacological agents as potential therapeutics for addiction.

Unbiased KOR Agonists Disrupt Cognition in a KOR Dependent Manner
Mice were trained on the delayed alternation task to learn and acquire the behavior requiring working memory functionality. The mice were divided into three groups, consisting of aCSF pretreated wildtype mice, norBNI pretreated wildtype mice, and PFC lox KOR mice, which lacked postsynaptic KOR expression in the PFC. These mice were then treated with saline and then performed the delayed alternation task. On the subsequent day, they were then treated with U50,488 and then performed the delayed alternation task. For all the trials, the percent of correct responses and total number of responses were recorded.

We observed that in the early phase of the delayed alternation task, mice pretreated with aCSF showed decrease performance on the task as well as decreased locomotor activity, as illustrated by the percent correct and total number of responses, respectively. However, mice that had been pretreated with norBNI or lacked KOR in the first place did not express this disruption (**Figure 27A**).

In the late phase of the task, the mice pretreated with aCSF continued to show decreased performance in the task. However, the lox KOR group also showed decreased performance. In the late phase, there appeared to be no change in locomotor activity as measured by total number of responses (**Figure 27B**).

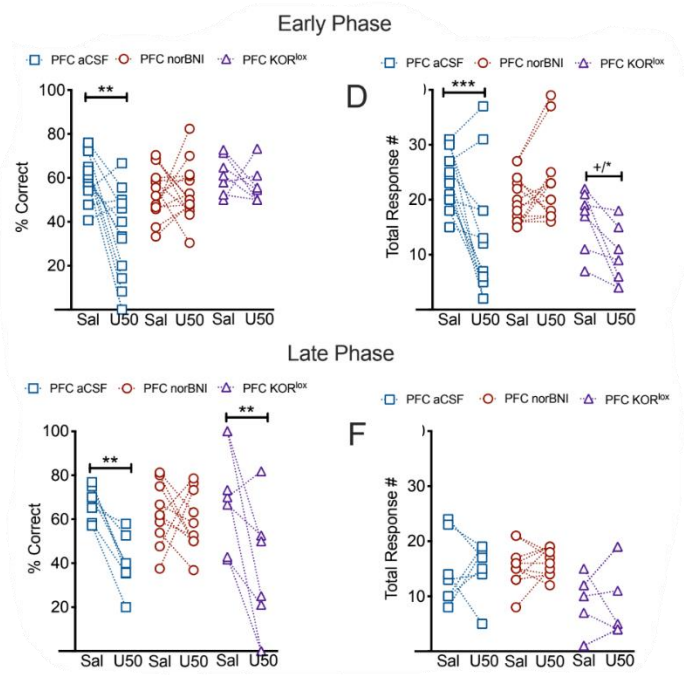


Figure 27: Conditional KOR knockouts in the PFC show no cognitive disruption during the early phase of delayed alternation task, but show disruption during the late phase. *Control mice treated with aCSF show cognitive disruption after injection of unbiased KOR agonist U50,488 in both early and late phases of the delayed alternation task. Mice pretreated with norBNI show no decrease in performance, suggesting that KOR in the PFC is involved in mediating cognitive disruptions. Mice lacking post-synaptic KOR show no cognitive disruption during the early phase of the task but show disruption during the late phase (Abraham, Casello, Wong et al., in preparation).*

Types of Stressors Differentially Activate KOR Circuit

After observing the effect of an unbiased KOR agonist on working memory, we tested different stressors to visualize their effect on KOR activation. By using KORp immunostaining, we were able to visualize the fraction of active receptors due to various stressors. We found that certain stressors were sufficient to induce KOR activation, while other stressors lacked the same potency. Most notably, we observed that the long-acting KOR antagonist norBNI effectively blocked KOR activation induced by stressors such as unbiased KOR agonists, morphine withdrawal, and optogenetic stimulation of dynorphin containing neurons in the PFC. Additionally, we observed that swim stress induced KOR activation in the DRN but not in the PFC (**Figure 28**).

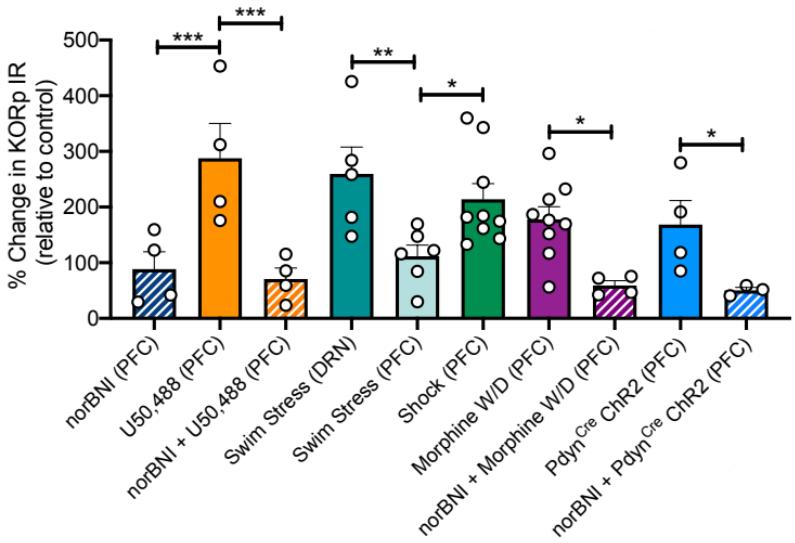


Figure 28: Different stressors are capable of evoking KOR activation. *Certain stressors show an increase in KORp immunoreactivity, which indicates activation of post-synaptic KOR. Pharmacological agents such as U50,488 are capable of activating KOR, which is blocked with pretreatment of norBNI. Environmental stressors such as shock and morphine withdrawal are capable of inducing KOR activation in the PFC. Swim stress was shown to activate KOR in the DRN, but not the PFC. Finally, optogenetic stimulation of dynorphin containing neurons in the PFC was shown to induce KOR activation, which was blocked with pretreatment with norBNI (Abraham et al., in preparation).*

Select Stressors Disrupt Working Memory in Mice

After observing the effects of an unbiased KOR agonist on the KOR circuit and confirming the KOR inducible nature of various stressors, we investigated the effects of those stressors on working memory. Mice were trained on the delayed alternation task and were split into different groups to investigate the effect of morphine withdrawal and optogenetic stimulation of dynorphin containing neurons in the PFC.

Wildtype mice and PFC lox KOR mice were trained on the delayed alternation task during a period of no treatment. They received a repeated morphine dosing procedure, followed by naloxone precipitated withdrawal. It was observed that both groups showed decrease performance in the delayed alternation task (**Figure 29A**). Additionally, mice expressing ChR2 in dynorphin-containing neurons were stimulated in the PFC to observe the effect of activation of dynorphin-containing neurons on working memory. It was observed that the stimulation resulted in a decrease in performance as measured by percent of correct alternations (**Figure 29B**).

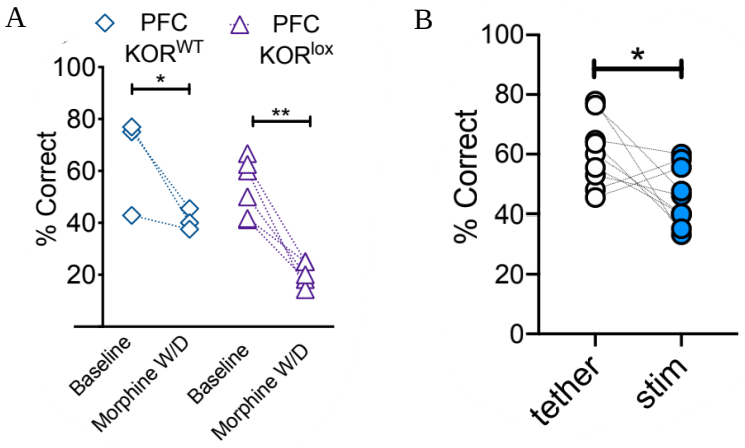


Figure 29: Morphine withdrawal and optogenetic stimulation of dynorphin containing neurons in the PFC are sufficient to disrupt cognition and working memory. *(A) For both wildtype and PFC lox KOR mice, morphine withdrawal causes a decrease in cognitive performance when completing the delayed alternation task. (B) Optogenetic stimulation of dynorphin containing neurons in the PFC also induces a decrease in cognitive performance in the delayed alternation task (Abraham, Casello, Wong et al., in preparation).*

Biased Agonists Affect Late Stage Cognition and Working Memory

Unbiased KOR agonists such as U50,488 suggest that activation of the KOR circuit results in decreased cognition and working memory. We investigated the effect of G protein biased KOR agonists such as nalfurafine on the delayed alternation task. In this experiment, mice were divided into two groups, consisting of PFC wildtype mice and PFC lox KOR mice. Each group was treated with both saline and nalfurafine to observe the differences in response. It was observed that nalfurafine administration resulted in decreased performance in the late stage of the task in both wildtype PFC mice and PFC lox KOR mice (**Figure 30**).

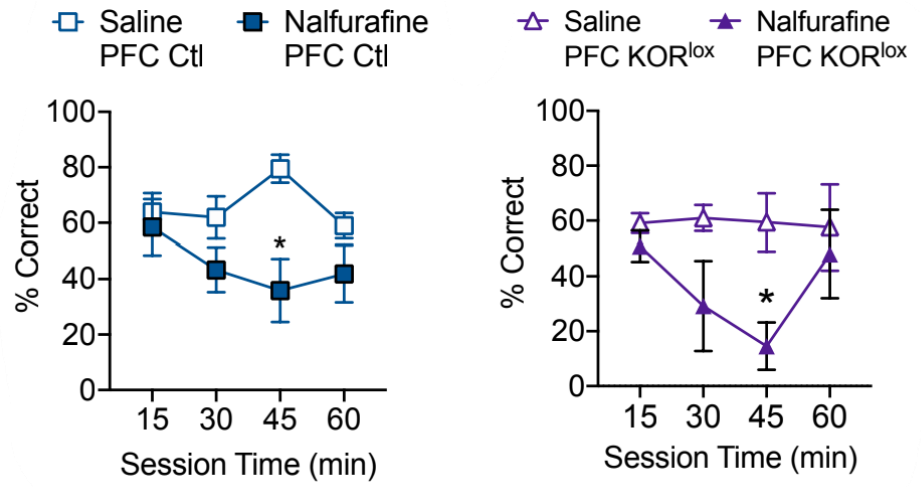


Figure 30: Nalfurafine induces late phase disruptions in cognitive performance for both wildtype and PFC lox KOR mice. *(Left) Wildtype mice injected with nalfurafine in the PFC demonstrate a disruption in cognitive performance only in the late phase of the delayed alternation task. (Right) Similarly, mice with post-synaptic KOR deleted also show an increase in cognitive disruption during the late phase of the delayed alternation task when compared with saline (Abraham, Casello, Wong et al., in preparation).*

5 Discussion

5.1 Section Overview

This section provides the interpretations of the observed results in this thesis. We demonstrate the validation of the genetically encoded sensors that were used and show how they generate useful insights at the cellular, neural circuit, and behavioral levels, and how they can inform future studies in animal models.

In this thesis, we developed a better understanding of how stress can affect and disrupt behavioral outputs such as motivation for drug seeking and cognition. Moreover, we investigated the role of KOR activation mediated by G protein and arrestin-dependent signaling pathways in modulating those behavioral outputs. All of this was performed with the use of genetically encoded sensors and optogenetic tools that were developed and validated with preliminary experiments that generated data which was computationally analyzed to determine the efficacy of those sensors.

5.2 Main Findings

Previous studies into KOR signaling cascades and dynorphin actions have been limited because there were limited tools for visualizing and quantifying activation of specific signaling pathways. Here we find that various genetically encoded sensors are useful in detecting chemical species or interactions of interest that provide useful insights into neural circuitry in general, and specifically the KOR-dynorphin circuit.

By performing single cell imaging on both *ex vivo* and HEK cells, we were able to determine potential sensor candidates for use in *in vivo* systems. We validated the use of the HyPerRed sensor for detecting concentrations of reactive oxygen species by imaging *ex vivo* brain slices that had been treated with nalfurafine. Additionally, we validated the use of the HyPerRed, Hy46, and HRM63 sensors in HEK cells and observed sensitive responses to KOR agonists from some, but not all, of the sensors.

Following single cell imaging, we performed validation of the sensors *in vivo*. We validated the use of HyPerRed, HRM63, and kLight by using fiber photometry. We found that U50,488 did not activate HyPerRed, while nalfurafine did. When tested with nalfurafine, we also found that kLight exhibited little fluorescent response. These results suggest that HyPerRed, and not kLight, would be a useful sensor for observing KOR mediated G protein activation and KOR expression in general. However, kLight allowed us to determine the temporal dynamics for KOR activation in response to different unbiased agonists, specifically stressors evoking dynorphin release. Thus, these results suggest the utility of kLight, and not HyPerRed, as a dynorphin sensor. These useful insights ultimately informed how we should utilize these sensors when performing behavioral experiments, and how kLight and HyPerRed could be used in tandem to uncover various aspects of the neural circuitry in question. These development and validations stages are summarized in **Figure 31.**

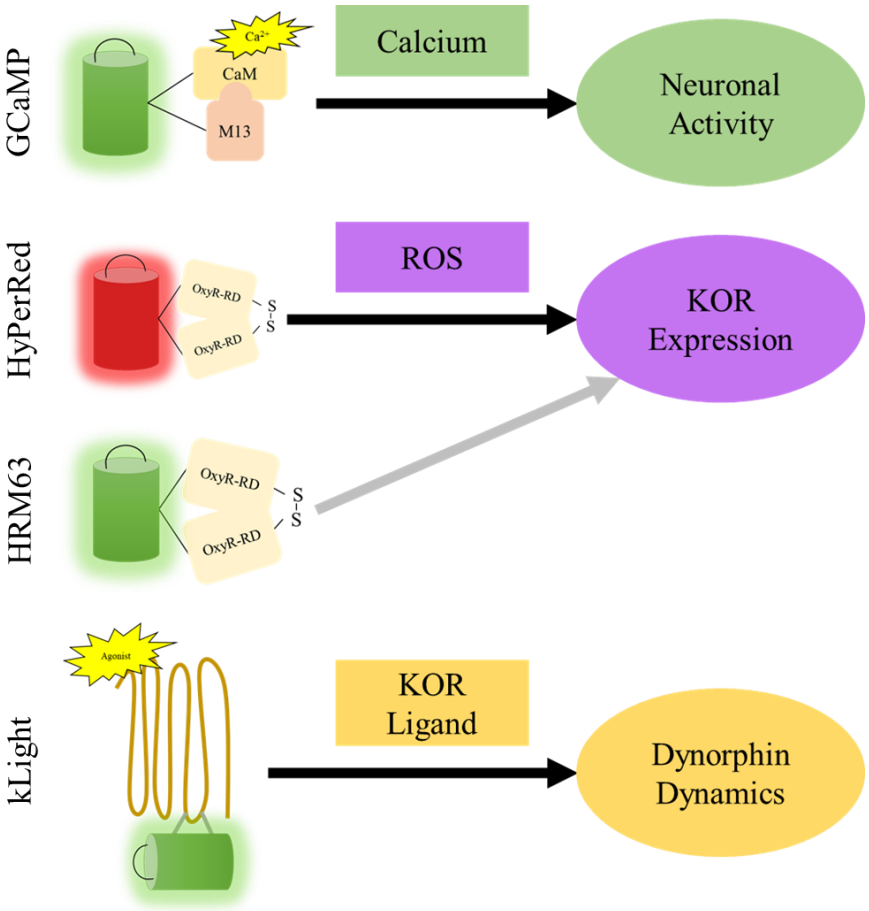


Figure 31: Summary of potential applications for validated sensors. *Each sensor studied in this thesis can provide useful information about neural circuits involved with stress and addiction. Black arrows indicate sensors that are sufficiently optimized for use in behavioral studies while gray arrows indicate sensors that require further optimization. While GCaMP is a useful sensor for detecting overall neuronal activity, it cannot provide information specific to KOR expression or dynorphin release. HyPerRed indicates the activation of KOR mediated G protein pathways and KOR expression but does not provide information about dynorphin dynamics. KLight can indicate the timing and location of dynorphin release but does not provide information about KOR expression. Together, these sensors can provide a more complete picture of how these various neural components interact with each other and influence addiction circuits.*

Integrating fiber photometry with a cocaine conditioned place preference experiment and repeated force swim stress exposure, we sought to better understand the role of KOR in dopaminergic VTA cells. Specifically, we investigated how stress affects these neurons to produce changes in neuronal activity and how these populations of neurons respond to the induction and relief of stress. We

found that by knocking out KOR on dopaminergic neurons in the VTA, CPP potentiation and increases in calcium transients were blocked. Additionally, we linked changes in neuronal activity in dopaminergic KOR expressing neurons with stress.

Finally, we investigated the presence of pre- and post-synaptic KOR and how they correlate to early and late phase cognitive disruptions. Additionally, we sought to understand stressors that were capable of evoking dynorphin release in the PFC and how stress can be a contributing factor to cognitive disruptions. We found evidence suggesting that pre-synaptic KOR is likely responsible for late phase disruptions, mediated through G protein dependent pathways, while post-synaptic KOR is likely responsible for early phase disruptions, mediated through an arrestin-dependent pathway.

Taken together, these findings suggest that a toolbox of novel genetically encoded sensors can be custom selected for each scientific question of interest and to generate useful insights that would be otherwise inaccessible.

5.3 Cellular

Population level responses to pharmacological agents or stressors can be distinct from single-cell responses and can mask important features highlighted by those higher resolution analyses. Since there is variability in composition from cell to cell, each one responds in a slightly different time course that may not be seen at the population level. Computationally, these findings are significant, as they allow for automation of the task of identifying regions of interest in single cell recordings. This automation allows for robust methods for identifying regions of interest that can save time compared with circling regions by hand. It also combines the identification and quantification steps so that they can be performed by a single program, rather than being performed in separate steps, thereby streamlining the process for determining single cell fluorescence. These techniques applied to both *ex vivo* and HEK cells allows for effective development and validation of genetically encoded sensors.

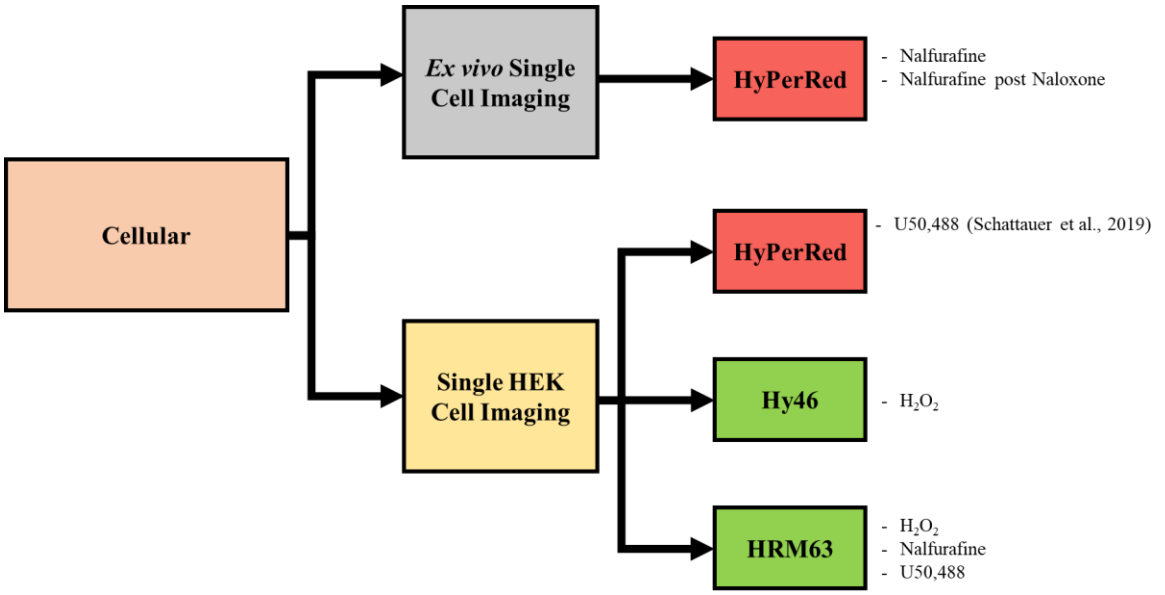


Figure 32: Summary of cellular experiments performed to develop and validate sensors. *Ex vivo single cell imaging was performed to validate the use of HyPerRed by using nalfurafine and nalfurafine post naloxone pretreatment. Single HEK cell imaging was used to validate HyPerRed, Hy46, and HRM63 in an in vivo setting. HyPerRed has previously been established to be responsive to KOR agonists such as U50,488, while Hy46 and HRM63 (HyPerRed green variants) were validated using H_2O_2 ramps and these KOR agonists.*

5.3.1 *Ex vivo* Single Cell Imaging Validation

Initial testing of the HyPerRed sensor shows that it is sensitive to reactive oxygen species generation from G protein biased agonists, such as nalfurafine. Since there is an increase in fluorescence for nalfurafine at a time course that is consistent with the increase in fluorescence in the HyPerRed fiber photometry, this suggests that HyPerRed is effectively able to report increases in reactive oxygen species that result from KOR activation through the G protein signaling pathway. Additionally, since this increase is blocked when pretreated with naloxone, this further reiterates the idea that HyPerRed can detect G protein mediated KOR activation.

Not only do these experiments illustrate the use of HyPerRed as an indicator for G protein mediated KOR activation, but they also suggest that individual cells in the VTA expressing KOR react at similar time courses. Since there was little variation between time to peak for the cells, this suggests that KOR expressing cells in the VTA act synchronously and can be modeled well by whole frame fluorescence.

These findings suggest that HyPerRed would be a suitable sensor for use in *in vivo* settings and in behavioral experiments for studying the effects of G protein biased agonists. However, there are limitations, since these trials were performed *ex vivo* and the HyPerRed responses could be different in a physiological setting, where other pathways could also play a role in reactive oxygen species generation that might not be accounted for in this model.

While the computational techniques that were designed can relatively accurately determine individual cells, there were still limitations with these techniques. Due to how the depth of field and how the images were captured, there are some cells that are out of focus. These cells are difficult to properly identify, and the program is unable to capture all of them. Therefore, further optimizations could be performed to manipulate the images to bring those cells back in focus.

5.3.2 Single HEK Cell Imaging Validation

Compared with the *ex vivo* single cell imaging, HEK cell imaging provides certain benefits, since these trials are performed in living cells. However, this presents other limitations and challenges, both when computationally processing and analyzing the data as well as limitations regarding considering other relevant pathways in an *in vivo* system. Even so, single HEK cell imaging is a cost-effective method that can still provide useful insights as to the feasibility of adapting certain sensors in an *in vivo* system.

Previous work has shown that U50,488 is sufficient to induce increases in fluorescence for the HyPerRed sensor (Schattauer et al., 2019). However, we sought to validate the use of the HyPerRed variant HRM63 as a reactive oxygen species sensor in a similar manner. To do so, we began with an HRM63 precursor, Hy46, to observe if the variant was capable of fluorescing under the presence of reactive oxygen species, after which we tested HRM63 itself.

In the case with the live cell imaging of HEK cells, we observed variability in the single cell response to pharmacological treatments consisting of H_2O_2, U50,488, and nalfurafine with different sensors. In this instance, both whole field and single-cell imaging was useful in providing relevant information regarding the time course of reactive oxygen species generation. Initial testing with the H_2O_2 ramp showed that the Hy46 sensor was capable of detecting changes in and sensitive to reactive oxygen species concentration. Individual cell traces highlighted the slight variability between intensity and timing of fluorescence that was otherwise masked in the population level traces.

The testing of HEK cells with the HRM63 sensor showed that H_2O_2 could induce a fluorescent response compared with vehicle. However, when tested with relevant KOR agonists such as U50,488 or nalfurafine, there was no observable response in the fluorescence levels. While the individual cells in the recordings were identified, there was no increase in fluorescence as would be expected from KOR activation. This suggests that, while capable of reporting the presence of reactive oxygen species when the reactive oxygen species are directly applied, there are other underlying issues in an *in vivo* setting with HRM63 expression or reporting that need to be optimized further.

While the analysis algorithm was able to track cells across the recording, it was less optimal at differentiating borders between cells that were in contact with each other. A correction for this was attempted through use of the watershedding technique, but the problem persisted in some of the frames. Thus, further optimizations could be performed to better identify borders between cells that are in contact to properly differentiate between such cells.

5.4 Neural Circuitry

While single cell resolution imaging provides some insights into the validity for the use of these genetically encoded sensors, we sought to further validate the sensors by using them *in vivo* through a fiber photometry system. We found that existing sensors such as GCaMP and some novel sensors such as HyPerRed and kLight can be used to probe signaling pathways involved with the KOR-dynorphin circuit. These findings provide preliminary evidence that the use of such sensors could be integrated in future experiments that would ultimately generate insights into unclear signaling pathways involving KOR and dynorphin.

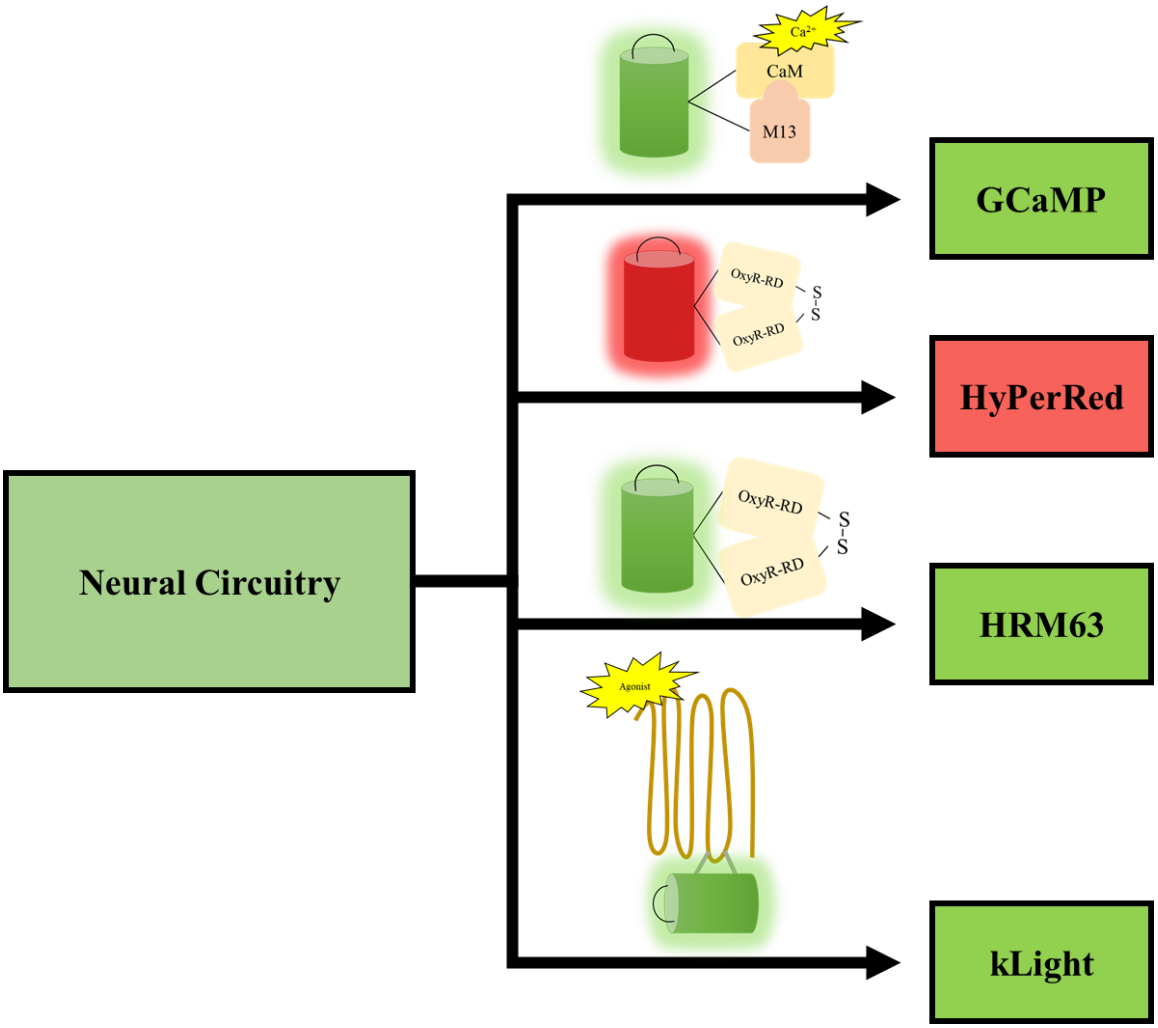


Figure 33: Summary of neural circuitry experiments performed to develop and validate sensors. *Initial validation steps involved using GCaMP in a CPP experiment to observe changes in fluorescence as a function of calcium concentration. Novel sensors such as HyPerRed, HRM63, and kLight were tested using various pharmacological treatments with biased and unbiased KOR agonists and other stressors inducing dynorphin release.*

5.4.1 Validation of Genetically Encoded Sensors

Experiments using fiber photometry were necessary for the development and validation of computational methods for analysis of novel sensors. While GCaMP has been an established method for detecting calcium concentrations and transients, other sensors could provide other useful insights regarding activated pathways of different KOR agonists. We found that other genetically encoded sensors could be utilized in a similar manner to GCaMP to be able to investigate neural circuitry involved in stress and drug seeking behaviors as well as investigating the effects of various KOR agonists.

Through use of genetically encoded reactive oxygen species sensors, we saw G protein mediated increases in reactive oxygen species concentrations as a result of KOR activation with G protein biased agonists. In combination with downstream G protein inhibitors such as CMPD101 and KOR antagonists such as naloxone, we were able to observe an inhibition of reactive oxygen species generation. This suggests that KOR mediated G protein pathways are responsible for the generation of reactive oxygen species and can also be used as a model for postsynaptic KOR activation.

Further validation was performed using the HyPerRed variant, HRM63. This sensor provided similar information as HyPerRed, with the main difference being through the green fluorescence in place of red fluorescence in HyPerRed.

Finally, a different class of sensor was used to determine dynorphin release through fluorescent signal. Mice expressing the kLight1.3 sensor were injected with differing dosages of U50,488, nalfurafine, and naloxone to observe the fluorescence response due to biased and unbiased KOR agonists with or without antagonists. We observed an increase in fluorescence for the U50,488 treatment which was absent in saline. This is reasonable, since U50,488 is an unbiased agonist, and binding of an agonist to kLight would result in fluorescence of the sensor. Furthermore, we found that mice treated with increasing dosages of U50,488 resulted in an increasing maximum fluorescence. By pretreatment with a non-selective opioid receptor antagonist such as naloxone, we found that fluorescence from kLight was blocked. These results suggest that kLight serves as an effective indicator for the presence of pharmacological unbiased KOR agonists.

To support these conclusions, we tested if naloxone induced withdrawal could also evoke kLight fluorescence. Chronic stressors have been shown to induce dynorphin release, which is an endogenous unbiased KOR agonist (Land et al., 2008). Additionally, studies have shown that KOR antagonists in the NAc can reduce the depressive-like behaviors resulting from morphine withdrawal (Zan et al., 2015) and that dynorphin could play a role in withdrawal from drugs of abuse (Chartoff and Carlezon, 2014). These preliminary studies suggest that morphine withdrawal should be able to evoke dynorphin release and thereby evoke kLight fluorescence. Our results show that kLight fluorescence from naloxone precipitated withdrawal causes an increase in fluorescence similar to levels of fluorescence from treatment with 10 mg/kg of U50,488. These findings suggest the utility of kLight as a dynorphin sensor, which would generate useful insights into the temporal and spatial dynamics of dynorphin release from different types of stressors.

Additionally, we tested the ability of biased KOR agonists to evoke a kLight response. We injected mice with nalfurafine, and we observed no difference with low or high doses of nalfurafine compared with saline. Since biased agonists activate KOR differently than unbiased agonists, this could

result from differences in conformational changes. Since kLight fluoresces due to KOR conformational changes, the change induced by biased agonists may not be specific enough for evoking a fluorescence response. However, further studies are needed to investigate how unbiased and biased KOR agonists evoke different kLight responses, which could be carried out by using other biased KOR agonists.

By developing and validating the use of these sensors *in vivo*, we prime them for future use in investigational studies into the preferential activation of different KOR-mediated pathways. Each sensor has specific advantages and limitations that govern their use in experimental procedures. Based on the scientific question of interest, these sensors can provide useful insights into the action of pharmacological agents and other stressors that could result in KOR activation. The trials further illustrate that these sensors can be used in animal models and provide information into which pathways are activated by biased or unbiased KOR activation, specific time courses for KOR activation, and connections within neural circuits that are important for mediating behavioral responses.

Comparing the sensors, these results suggest that U50,488 does not activate HyPerRed, but nalfurafine does. Therefore, by using HyPerRed, we can identify cell populations that are likely to contain KOR by infecting those populations with HyPerRed and observing the reaction to a KOR agonist such as nalfurafine. While HyPerRed provides information about the location of KOR, kLight can provide insights as to the temporal and spatial dynamics of dynorphin since it directly measures dynorphin binding. Since HyPerRed is a red sensor and kLight is a green sensor, the sensors could be included in a multiplexed experiment to further investigate KOR-dynorphin dynamics.

5.4.2 GCaMP Fiber Photometry

While previous studies have investigated KOR in dopaminergic VTA cells with *in vivo* electrophysiology and voltammetry, the dynamics of KOR action in these cell populations remained unclear. It is known that systemic administration of KOR antagonists block conditioned place preference potentiation, but it was less clear how dopaminergic neurons expressing KOR is also responsible for potentiation. This raised the question as to what stress is doing to dopaminergic neurons to produce changes in then neurocircuitry.

Previous studies have shown using DRN KOR knockouts that serotonergic neurons expressing KOR in the DRN are involved in conditioned place aversion (Land et al., 2009). Other studies have additionally shown that conditioned place aversion can be blocked using VTA KOR knockouts on dopaminergic neurons (Chefer et al., 2013; Ehrich et al., 2015). However, it remains unclear how stress and dynorphin interacts with dopaminergic KOR in the VTA and their effect on behavior. Specifically, we investigated how stressful experiences influence reward behaviors through KOR expressing dopaminergic neurons in the VTA. It was seen that by knocking out KOR in the VTA, this blocked calcium signal decreases as a result of repeated force swim stress, as well as blocked calcium signal increases after relief of the repeated force swim stress. These results indicate that stressors can modulate dopaminergic neuronal activity in the VTA in a KOR dependent manner, and that by knocking out KOR in this population of neurons, it can block those disruptions. However, we were not able to observe a direct linkage between neuronal activity and behavior during

the conditioned place preference test. This could have been because the fiber photometry setup may have confounded certain aspects of the test. Despite this, however, these results suggest that there is a connection between stress potentiation and these populations of neurons.

5.5 Behavior

Our findings suggest that presynaptic and postsynaptic KOR play different roles in mediating KOR induced signaling cascades that result in early and late stage disruptions in the delayed alternation procedure.

When comparing between groups of mice that were pretreated with norBNI versus PFC lox KOR mice treated with a Cre recombinase AAV, both groups showed inhibition of cognitive disruption following KOR activation with an unbiased agonist during the early phase of the delayed alternation task. However, only the norBNI pretreated group exhibited further inhibition of cognitive disruption during the late phase of the procedure. Since the Cre-recombinase procedure deletes postsynaptic KOR and norBNI acts as an antagonist for both pre and postsynaptic KOR, this suggests that presynaptic KOR are responsible for late phase disruptions and postsynaptic KOR are responsible for early phase disruptions.

Furthermore, our results indicate that KORp staining shows arrestin-dependent postsynaptic activation of KOR, and that certain stressors that cause behavioral disruption also induce postsynaptic activation of KOR. Testing stressors such as morphine withdrawal and optogenetic stimulation of PFC dynorphin containing neurons observe a late phase disruption in cognition and working memory, suggesting that these effects might be presynaptic since the PFC lox KOR mice also observe a similar disruption, but more testing is needed to confirm this. While KORp staining provides information about activation of the arrestin-dependent pathway, this technique does not provide information about G protein-dependent post synaptic activation. Thus, the development and validation of the HyPerRed sensor was important to probe the G protein KOR mediated activation.

In the HyPerRed fiber photometry experiments, it was observed that nalfurafine induced an increase in reactive oxygen species, which is thought to be produced through the G protein pathway. In a following delayed alternation procedure when either PFC lox KOR mice or wildtype mice were injected with nalfurafine, both groups exhibited cognitive disruption in the late phase of the delayed alternation procedure. This further reiterates the idea that presynaptic KOR is responsible for late phase disruptions, since both groups had presynaptic KOR and observed late phase disruptions. Contrasting with the HyPerRed nalfurafine results which indicate that there is, in fact, postsynaptic activation of KOR, this would suggest that KOR mediated G protein activation is not sufficient to induce behavioral disruptions in the early phase. However, since we observe both early and late phase behavioral disruption in aCSF pretreated mice after KOR activation, this would suggest that the arrestin-dependent pathway is responsible for the early phase disruptions.

Previous work has shown that most KOR in the PFC is presynaptic (Tejeda et al, 2015). In these studies, there was little electrophysiological response to postsynaptic KOR activation. This raises the question as to the role of postsynaptic KOR. Using the delayed alternation task, we demonstrated the presence and role of postsynaptic KOR in the PFC using KORp staining, PFC lox KOR mice, and the HyPerRed reactive oxygen species sensor. Each of these methods indicate the presence of postsynaptic KOR.

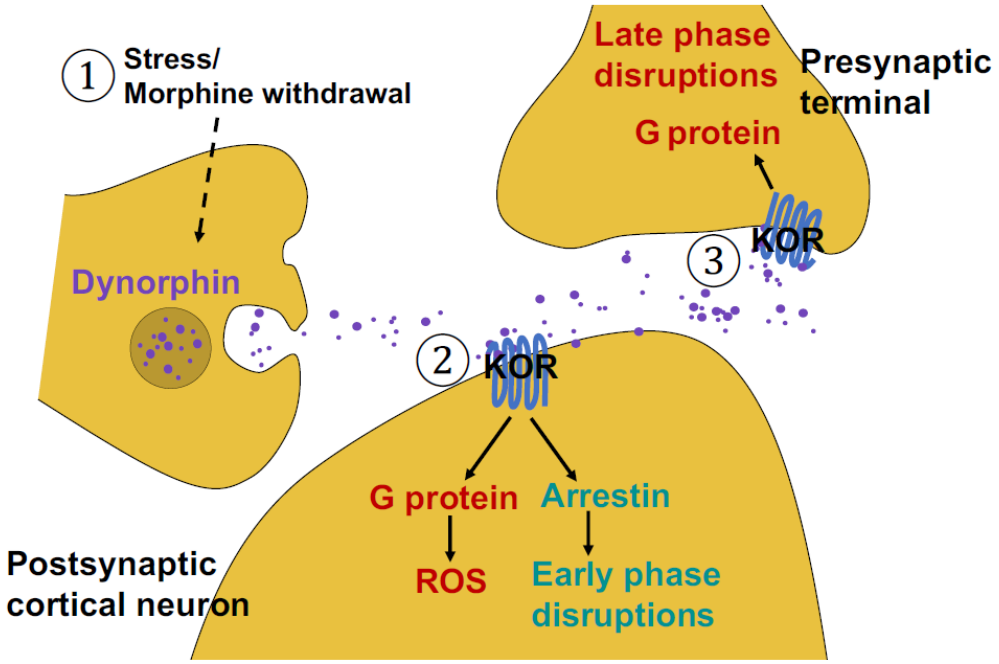


Figure 34: Pre- and postsynaptic KOR mediate early and late phase cognitive disruptions. *(1) Certain stressors mediate dynorphin release onto both pre- and postsynaptic KOR in the PFC. (2) Postsynaptic KOR is likely responsible for early phase cognitive disruptions through arrestin-dependent pathways, while postsynaptic KOR activation can be measured with HyPerRed through reactive oxygen species generation. (3) Presynaptic KOR is likely responsible for late phase disruptions through G protein pathways (Abraham, Casello, Wong et al., in preparation).*

These results respond in part to the question as to the use of KOR antagonists as potential pharmacological treatments. While KOR antagonists can be useful for motivational and affective changes, our results suggest that they may also be useful for inhibiting cognitive disruptions and that KOR antagonists can block some aspects of stress induced disruptions in working memory. These results work toward disentangling the KOR-dynorphin circuit and using the tools that we have developed to better understand those dynamics.

5.6 Future Directions

While the objectives of this thesis were able to develop and validate the use of these genetically encoded sensors, more experiments could be performed to further the development of these sensors. For example, we were unable to visualize fluorescence *in vivo* on a single-cell resolution. Tools such as the UCLA Miniscope project or other miniature head-mounted Miniscopes could be adapted to be used in animal models (Aharoni et al., 2018). These tests would validate the use of such sensors and would provide useful insights into the live single-cell dynamic responses to pharmacological agents and stressors while retaining appropriate physiological conditions, in contrast with either the *ex vivo* or single HEK cell imaging.

The UCLA Miniscope could be assembled and used in similar experiments that test the various effects of pharmacological agents and stressors in combination with the developed genetically encoded sensors. This head-mounted miniscope would be a logical next step, as it allows for behavioral assays in freely awake and behaving mice.

To further integrate both neural circuitry and behavior, future experiments could be performed by using fiber photometry together with delayed alternation. Such an experiment would provide an increased level of resolution for the time course of KOR activation and could provide more complete insights as to the timing of pre and postsynaptic KOR activation. These experiments could be performed with the sensors that were developed, namely HyPerRed, HRM63, or kLight1.3.

Further fiber photometry experiments could be run using kLight and HyPerRed together in a multiplexed experiment. Such an experiment would generate insights as to the action of dynorphin as well as KOR expression. Since kLight indicates the interaction of a ligand, such as dynorphin, with KOR, it could clearly indicate when dynorphin is released. Additionally, HyPerRed could indicate the regions that express KOR and are thus activated by dynorphin.

To follow up on the findings of the delayed alternation task and the role of pre and postsynaptic KOR in cognitive function, the early phase disruption of cognitive function could be investigated by using an arrestin-biased KOR agonist. However, since such an agonist does not currently exist, these tests would not be possible at the present time.

Additionally, computational methods such as machine learning could be applied to behavioral data to quantify and understand behavioral responses to pharmacological treatments and stressors. Using the tool DeepLabCut, developed by Mathis et al, we could mark specific responses from the mice to characterize behaviors such as morphine withdrawal (Mathis et al., 2018). This data could provide useful insights that would link cellular and neural circuit responses with behavioral outputs.

5.7 New Understanding

These findings illustrate the value of various optogenetic sensors for use in animal models. As these sensors have been created in recent years, much of how they interact at a physiological level, their sensitivity to their respective reporters, and capability of indicating specific pathways that result from KOR activation in animal models still required further investigation. Through the experiments performed, we investigated those dynamics and prepared the further use of those sensors in animal studies, with the goal to probe certain aspects of the KOR-dynorphin circuit. A summary for each sensor that was used for these experiments, along with the main advantages and limitations for using each sensor are described in **Table 6**.

Table 6: Table summarizing new understanding regarding sensors that were developed and validated. *Each sensor has specific advantages and limitations that provide unique insights regarding KOR-dynorphin circuit dynamics.*

Sensor	Detection	Advantages	Limitations
GCaMP	Ca^{2+}	- Commonly used and therefore optimized - Complementary to electrophysiology measurements	- Does not measure neuronal activity directly, only through calcium concentration - May not provide insight into effects of biased agonism
HyPerRed	ROS	- Red fluorescence allows for multiplexing experiments - Provides insights into effects of G protein biased agonism	- Lower sensitivity as compared with more established sensors - Other pathways resulting in ROS generation could influence fluorescence
HyPerRed Variants	ROS	- Green fluorescence - Differing sensitivity from HyPerRed	- Other pathways resulting in ROS generation could influence fluorescence - Not currently optimized for *in vivo* recordings
kLight1.3	KOR Ligand	- Capable of directly measuring KOR activation - Provides insight into temporal dynamics of dynorphin	- Cannot provide insight into effects of biased agonism - Novel sensor, could be further optimized

After the validation, we then used the insights gained by using the sensors to inform how certain pathways affect behavioral outputs. Future studies that will be performed can be informed by the results of this thesis as to which sensor will provide the necessary readout to investigate the scientific question of interest.

5.8 Conclusions

These experiments have demonstrated the value and validity of using optogenetic tools such as genetically encoded sensors for use in animal models. Initial tests at single cell resolution have shown that some of these sensors can detect certain chemical species of interest and reporting their presence with sufficient sensitivity, and that other sensors require further optimization before they can be adapted for use in animal models. These tests were at a cellular resolution that was performed in both *ex vivo* and *in vivo* systems as a cost-effective way to preliminarily determine the feasibility

of adapting these sensors to be used in animal models. Additionally, computational methods were adapted and designed to handle these datasets and automate the procedures involved with their analyses.

Furthermore, the novel genetically encoded sensors were tested and validated in animal models through fiber photometry, which allowed us to visualize neuronal activity and KOR mediated signaling pathways. Computational methods were developed to analyze these large datasets and extract useful information from the fluorescence recordings. These methods allowed us to determine if the sensors had the necessary sensitivity for reporting the chemical species of interest, which included calcium, reactive oxygen species, or receptor ligands.

Finally, we used the insights gained from the validation and development stages for these sensors to better understand the neurocircuitry in different brain regions involved with stress and behavior. These tests have important clinical relevance because they work towards the development of either KOR agonists or antagonists as potential therapeutic alternatives to addictive substances such as morphine. Pharmacological agents that could replace morphine and other addictive opioids would aim to attenuate the strain felt by the current opioid epidemic and ultimately contribute to a safer, healthier, and more productive society.

5.9 Acknowledgements

5.9.1 Key Personnel

I would like to primarily acknowledge Dr. Charles Chavkin as my principal investigator for his guidance throughout my academic career and for allowing me the resources to perform work in his lab. I would also like to especially thank Dr. Antony Abraham as my personal lab mentor, for taking me on as a mentee, allowing me the opportunity to take part in exciting and impactful research, for his exceptional patience and kindness, and for his dedication to the academic success of each of his students. Additionally, I would like to thank Dr. Benjamin Land for his mentorship and advice in my work in the lab. I would like to thank Dr. Selena Schattauer and Katie Reichard for allowing me to contribute to their experiments with single cell imaging. Finally, I would like to thank all other members of the Chavkin Lab for their support throughout my time in the lab and Dr. Andre Berndt for being willing to participate in my supervisory committee.

5.9.2 Equipment, Facilities, and Software

All experiments were designed and performed in the Chavkin Lab spaces in the University of Washington Health Sciences building. All computational analyses were performed using a personal educational license of MATLAB version 2015a. All statistical analyses were performed using Prism 8. EthoVision by Noldus was used for analysis of select behavioral experiments.

5.9.3 Funding

I would like to thank the Addiction Center for Excellence for partially funding the materials and spaces for this work. (P50-MH106428 (CC), P30-DA048736 (CC), R01-DA030074 (CC). The author declares no financial conflict of interest).

6 References

Abraham, Antony D, et al. “Prefrontocortical Dynorphin Opioids Disrupt Cognition through Distinct Pre- and Postsynaptic Mechanisms.” 2020.

Abraham, Antony D, et al. “κ-Opioid Receptor Activation in Dopamine Neurons Disrupts Behavioral Inhibition.” *Neuropsychopharmacology*, vol. 43, no. 2, 2017, pp. 362–372., doi:10.1038/npp.2017.133.

Abraham, Antony D., et al. “Orally Consumed Cannabinoids Provide Long-Lasting Relief of Allodynia in a Mouse Model of Chronic Neuropathic Pain.” *Neuropsychopharmacology*, 2019, doi:10.1038/s41386-019-0585-3.

Aharoni, Daniel, et al. “All the Light That We Can See: a New Era in Miniaturized Microscopy.” *Nature Methods*, vol. 16, no. 1, 2018, pp. 11–13., doi:10.1038/s41592-018-0266-x.

Allouche, Stéphane, et al. “Opioid Receptor Desensitization: Mechanisms and Its Link to Tolerance.” *Frontiers in Pharmacology*, vol. 5, 2014, doi:10.3389/fphar.2014.00280.

Alvarez, Veronica A., et al. “μ-Opioid Receptors: Ligand-Dependent Activation of Potassium Conductance, Desensitization, and Internalization.” *The Journal of Neuroscience*, vol. 22, no. 13, 2002, pp. 5769–5776., doi:10.1523/jneurosci.22-13-05769.2002.

Andrews, Mackenzie M. “Physical and Computational Methods of Investigating the Relationship between Stress, Cognition, and Behavior in the Context of the KOR-Dynorphin System.” *University of Washington*, 2019.

Ballard, T. M., et al. “Effect of a Thyrotrophin-Releasing Hormone Analogue, RX77368, on AMPA-Induced Septal-Hippocampal Lesioned Rats in an Operant Delayed Non-Matching to Position Test.” *Psychopharmacology*, vol. 127, no. 1-2, 1996, pp. 265–275., doi:10.1007/bf02806002.

Belousov, Vsevolod V, et al. “Genetically Encoded Fluorescent Indicator for Intracellular Hydrogen Peroxide.” *Nature Methods*, vol. 3, no. 4, 2006, pp. 281–286., doi:10.1038/nmeth866.

Bidlack, Jean M. “Mixed Kappa/Mu Partial Opioid Agonists as Potential Treatments for Cocaine Dependence.” *Emerging Targets & Therapeutics in the Treatment of Psychostimulant Abuse Advances in Pharmacology*, 2014, pp. 387–418., doi:10.1016/b978-0-12-420118-7.00010-x.

Bruchas, M. R., et al. “Stress-Induced p38 Mitogen-Activated Protein Kinase Activation Mediates Kappa-Opioid-Dependent Dysphoria.” *Journal of Neuroscience*, vol. 27, no. 43, 2007, pp. 11614–11623., doi:10.1523/jneurosci.3769-07.2007.

Bruchas, Michael R., and Charles Chavkin. “Kinase Cascades and Ligand-Directed Signaling at the Kappa Opioid Receptor.” *Psychopharmacology*, vol. 210, no. 2, 2010, pp. 137–147., doi:10.1007/s00213-010-1806-y.

Bruchas, Michael R., et al. “Kappa Opioid Receptor Activation of p38 MAPK Is GRK3- and Arrestin-Dependent in Neurons and Astrocytes.” *Journal of Biological Chemistry*, vol. 281, no. 26, 2006, pp. 18081–18089., doi:10.1074/jbc.m513640200.

Carlezon, William A., et al. “Regulation of Cocaine Reward by CREB.” *Science*, vol. 282, no. 5397, 1998, pp. 2272–2275., doi:10.1126/science.282.5397.2272.

Carlezon, William A., et al. “Depressive-Like Effects of the κ-Opioid Receptor Agonist Salvinorin A on Behavior and Neurochemistry in Rats.” *Journal of Pharmacology and Experimental Therapeutics*, vol. 316, no. 1, 2005, pp. 440–447., doi:10.1124/jpet.105.092304.

Chartoff, Elena H, et al. “Relative Timing Between Kappa Opioid Receptor Activation and Cocaine Determines the Impact on Reward and Dopamine Release.” *Neuropsychopharmacology*, vol. 41, no. 4, 2015, pp. 989–1002., doi:10.1038/npp.2015.226.

Chartoff, Elena H., and William A. Carlezon. “Drug Withdrawal Conceptualized as a Stressor.” *Behavioural Pharmacology*, vol. 25, no. 5 and 6, 2014, pp. 473–492., doi:10.1097/fbp.0000000000000080.

Chavkin, Charles, et al. “Dynorphin Is a Specific Endogenous Ligand of the Kappa Opioid Receptor.” *Science*, vol. 215, no. 4531, 1982, pp. 413–415., doi:10.1126/science.6120570.

Chefer, Vladimir I, et al. “Kappa Opioid Receptors on Dopaminergic Neurons Are Necessary for Kappa-Mediated Place Aversion.” *Neuropsychopharmacology*, vol. 38, no. 13, 2013, pp. 2623–2631., doi:10.1038/npp.2013.171.

Chiara, Gaetano Di. “Nucleus Accumbens Shell and Core Dopamine: Differential Role in Behavior and Addiction.” *Behavioural Brain Research*, vol. 137, no. 1-2, 2002, pp. 75–114., doi:10.1016/s0166-4328(02)00286-3.

Chorover, S. L., and C. G. Gross. “Caudate Nucleus Lesions: Behavioral Effects in the Rat.” *Science*, vol. 141, no. 3583, 1963, pp. 826–827., doi:10.1126/science.141.3583.826.

Conrad, Cheryl D. “A Critical Review of Chronic Stress Effects on Spatial Learning and Memory.” *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, vol. 34, no. 5, 2010, pp. 742–755., doi:10.1016/j.pnpbp.2009.11.003.

Coons, Albert H., et al. “The Demonstration of Pneumococcal Antigen in Tissues by the Use of Fluorescent Antibody.” *The Journal of Immunology*, vol. 45, no. 3, 1 Nov. 1942, pp. 159–170.

Davis, Michael, et al. “Phasic vs Sustained Fear in Rats and Humans: Role of the Extended Amygdala in Fear vs Anxiety.” *Neuropsychopharmacology*, vol. 35, no. 1, 2009, pp. 105–135., doi:10.1038/npp.2009.109.

Ehrich, J. M., et al. “Kappa Opioid Receptor-Induced Aversion Requires p38 MAPK Activation in VTA Dopamine Neurons.” *Journal of Neuroscience*, vol. 35, no. 37, 2015, pp. 12917–12931., doi:10.1523/jneurosci.2444-15.2015.

Ermakova, Yulia G., et al. “Red Fluorescent Genetically Encoded Indicator for Intracellular Hydrogen Peroxide.” *Nature Communications*, vol. 5, no. 1, 2014, doi:10.1038/ncomms6222.

Florence, Curtis S., et al. “The Economic Burden of Prescription Opioid Overdose, Abuse, and Dependence in the United States, 2013.” *Medical Care*, vol. 54, no. 10, 2016, pp. 901–906., doi:10.1097/mlr.0000000000000625.

Glick, Stanley D., et al. “Kappa Opioid Inhibition of Morphine and Cocaine Self-Administration in Rats.” *Brain Research*, vol. 681, no. 1-2, 1995, pp. 147–152., doi:10.1016/0006-8993(95)00306-b.

Goeders, N. “The Impact of Stress on Addiction.” *European Neuropsychopharmacology*, vol. 13, 2003, doi:10.1016/s0924-977x(03)90003-4.

Gold, Phillip W., et al. “Clinical and Biochemical Manifestations of Depression: Relation to the Neurobiology of Stress.” *Neural Plasticity*, vol. 2015, 2015, pp. 1–11., doi:10.1155/2015/581976.

Green, Matthew R., and Cheryl M. McCormick. “Effects of Social Instability Stress in Adolescence on Long-Term, Not Short-Term, Spatial Memory Performance.” *Behavioural Brain Research*, vol. 256, 2013, pp. 165–171., doi:10.1016/j.bbr.2013.08.011.

Gunaydin, Lisa A., et al. “Natural Neural Projection Dynamics Underlying Social Behavior.” *Cell*, vol. 157, no. 7, 2014, pp. 1535–1551., doi:10.1016/j.cell.2014.05.017.

Gutscher, Marcus, et al. “Proximity-Based Protein Thiol Oxidation by H2O2-Scavenging Peroxidases.” *Journal of Biological Chemistry*, vol. 284, no. 46, 2009, pp. 31532–31540., doi:10.1074/jbc.m109.059246.

Hamers, Danny, et al. “Development of FRET Biosensors for Mammalian and Plant Systems.” *Protoplasma*, vol. 251, no. 2, 2013, pp. 333–347., doi:10.1007/s00709-013-0590-z.

Jennings, Joshua H., et al. “Distinct Extended Amygdala Circuits for Divergent Motivational States.” *Nature*, vol. 496, no. 7444, 2013, pp. 224–228., doi:10.1038/nature12041.

Katz, R.j., et al. “Acute and Chronic Stress Effects on Open Field Activity in the Rat: Implications for a Model of Depression.” *Neuroscience & Biobehavioral Reviews*, vol. 5, no. 2, 1981, pp. 247–251., doi:10.1016/0149-7634(81)90005-1.

Kennedy, Nicole M., et al. “Optimization of a Series of Mu Opioid Receptor (MOR) Agonists with High G Protein Signaling Bias.” *Journal of Medicinal Chemistry*, vol. 61, no. 19, 2018, pp. 8895–8907., doi:10.1021/acs.jmedchem.8b01136.

Kiss, Alexi, et al. “Kappa Opioids Promote the Proliferation of Astrocytes via Gβγ and β-Arrestin 2-Dependent MAPK-Mediated Pathways.” *Journal of Neurochemistry*, vol. 107, no. 6, 2008, pp. 1753–1765., doi:10.1111/j.1471-4159.2008.05745.x.

Knoll, Allison T., and William A. Carlezon. “Dynorphin, Stress, and Depression.” *Brain Research*, vol. 1314, 2010, pp. 56–73., doi:10.1016/j.brainres.2009.09.074.

Koob, George F, and Nora D Volkow. “Neurocircuitry of Addiction.” *Neuropsychopharmacology*, vol. 35, no. 1, 2009, pp. 217–238., doi:10.1038/npp.2009.110.

Koob, George F. “Neural Mechanisms of Drug Reinforcement.” *Annals of the New York Academy of Sciences*, vol. 654, no. 1 The Neurobiol, 1992, pp. 171–191., doi:10.1111/j.1749-6632.1992.tb25966.x.

Koob, George F. “The Neurocircuitry of Addiction: Implications for Treatment.” *Clinical Neuroscience Research*, vol. 5, no. 2-4, 2005, pp. 89–101., doi:10.1016/j.cnr.2005.08.005.

Koob, George F., et al. “Addiction as a Stress Surfeit Disorder.” *Neuropharmacology*, vol. 76, 2014, pp. 370–382., doi:10.1016/j.neuropharm.2013.05.024.

Koob, George, and Mary Jeanne Kreek. “Stress, Dysregulation of Drug Reward Pathways, and the Transition to Drug Dependence.” *American Journal of Psychiatry*, vol. 164, no. 8, 2007, pp. 1149–1159., doi:10.1176/appi.ajp.2007.05030503.

Land, Benjamin B., et al. “The Dysphoric Component of Stress Is Encoded by Activation of the Dynorphin K-Opioid System.” *Journal of Neuroscience*, vol. 28, no. 2, 2008, pp. 407–414., doi:10.1523/jneurosci.4458-07.2008.

Land, Benjamin B., et al. “Activation of the Kappa Opioid Receptor in the Dorsal Raphe Nucleus Mediates the Aversive Effects of Stress and Reinstates Drug Seeking.” *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, 2009, pp. 19168–19173., doi:10.1073/pnas.0910705106.

Lefkowitz, Robert J., and Sudha K. Shenoy. “Transduction of Receptor Signals by -Arrestins.” *Science*, vol. 308, no. 5721, 2005, pp. 512–517., doi:10.1126/science.1109237.

Lerner, Talia N., et al. “Intact-Brain Analyses Reveal Distinct Information Carried by SNc Dopamine Subcircuits.” *Cell*, vol. 162, no. 3, 2015, pp. 635–647., doi:10.1016/j.cell.2015.07.014.

Locke, Timothy M., et al. “Dopamine D1 Receptor–Positive Neurons in the Lateral Nucleus of the Cerebellum Contribute to Cognitive Behavior.” *Biological Psychiatry*, vol. 84, no. 6, 2018, pp. 401–412., doi:10.1016/j.biopsych.2018.01.019.

Mague, Stephen D., et al. “Antidepressant-Like Effects of κ-Opioid Receptor Antagonists in the Forced Swim Test in Rats.” *Journal of Pharmacology and Experimental Therapeutics*, vol. 305, no. 1, 2003, pp. 323–330., doi:10.1124/jpet.102.046433.

Marton-Popovici, Monica, et al. “κ Opioid Receptor Antagonism and Prodynorphin Gene Disruption Block Stress-Induced Behavioral Responses.” *The Journal of Neuroscience*, vol. 23, no. 13, 2003, pp. 5674–5683., doi:10.1523/jneurosci.23-13-05674.2003.

Marx, Vivien. “Probes: FRET Sensor Design and Optimization.” *Nature Methods*, vol. 14, no. 10, 2017, pp. 949–953., doi:10.1038/nmeth.4434.

Mathis, Alexander, et al. “DeepLabCut: Markerless Pose Estimation of User-Defined Body Parts with Deep Learning.” *Nature Neuroscience*, vol. 21, no. 9, 2018, pp. 1281–1289., doi:10.1038/s41593-018-0209-y.

McEwen, Bruce S. “Protective and Damaging Effects of Mediators of Stress and Adaptation: Central Role of the Brain.” *Brain, Behavior, and Immunity*, vol. 25, 2011, doi:10.1016/j.bbi.2011.07.216.

Mclaughlin, Jay P, et al. “Prior Activation of Kappa Opioid Receptors by U50,488 Mimics Repeated Forced Swim Stress to Potentiate Cocaine Place Preference Conditioning.” *Neuropsychopharmacology*, vol. 31, no. 4, 2005, pp. 787–794., doi:10.1038/sj.npp.1300860.

McLaughlin, Jay P., et al. “κ Opioid Receptor Antagonism and Prodynorphin Gene Disruption Block Stress-Induced Behavioral Responses.” *The Journal of Neuroscience*, vol. 23, no. 13, 2003, pp. 5674–5683., doi:10.1523/jneurosci.23-13-05674.2003.

Miess, Elke, et al. “Multisite Phosphorylation Is Required for Sustained Interaction with GRKs and Arrestins during Rapid μ-Opioid Receptor Desensitization.” *Science Signaling*, vol. 11, no. 539, 2018, doi:10.1126/scisignal.aas9609.

Minneman, Kenneth P., and Leslie L. Iversen. “Enkephalin and Opiate Narcotics Increase Cyclic GMP Accumulation in Slices of Rat Neostriatum.” *Nature*, vol. 262, no. 5566, 1976, pp. 313–314., doi:10.1038/262313a0.

Nagai, T., et al. “Circularly Permuted Green Fluorescent Proteins Engineered to Sense Ca2+.” *Proceedings of the National Academy of Sciences*, vol. 98, no. 6, 2001, pp. 3197–3202., doi:10.1073/pnas.051636098.

Nakai, Junichi, et al. “A High Signal-to-Noise Ca2+ Probe Composed of a Single Green Fluorescent Protein.” *Nature Biotechnology*, vol. 19, no. 2, 2001, pp. 137–141., doi:10.1038/84397.

Nestler, Eric J., and William A. Carlezon. “The Mesolimbic Dopamine Reward Circuit in Depression.” *Biological Psychiatry*, vol. 59, no. 12, 2006, pp. 1151–1159., doi:10.1016/j.biopsych.2005.09.018.

Newton, Samuel S., et al. “Inhibition of CAMP Response Element-Binding Protein or Dynorphin in the Nucleus Accumbens Produces an Antidepressant-Like Effect.” *The Journal of Neuroscience*, vol. 22, no. 24, 2002, pp. 10883–10890., doi:10.1523/jneurosci.22-24-10883.2002.

Noble, Florence, et al. “Opioid Receptor Desensitization: Mechanisms and Its Link to Tolerance.” *Frontiers in Pharmacology*, vol. 5, 2014, doi:10.3389/fphar.2014.00280.

Olsen, Christopher M. “Natural Rewards, Neuroplasticity, and Non-Drug Addictions.” *Neuropharmacology*, vol. 61, no. 7, 2011, pp. 1109–1122., doi:10.1016/j.neuropharm.2011.03.010.

Owens, Michael J., and Charles B. Nemeroff. “Physiology and Pharmacology of Corticotropin-Releasing Factor.” *Pharmacological Reviews*, vol. 43, no. 4, 1991, pp. 426–462.

Patriarchi, Tommaso, et al. “Ultrafast Neuronal Imaging of Dopamine Dynamics with Designed Genetically Encoded Sensors.” *Science*, vol. 360, no. 6396, 2018, doi:10.1126/science.aat4422.

Piazza, Pier Vincenzo, et al. “Stress- and Pharmacologically-Induced Behavioral Sensitization Increases Vulnerability to Acquisition of Amphetamine Self-Administration.” *Brain Research*, vol. 514, no. 1, 1990, pp. 22–26., doi:10.1016/0006-8993(90)90431-a.

Reichard, Kathryn L., et al. “Regulation of Kappa Opioid Receptor Inactivation Depends on Sex and Cellular Site of Antagonist Action.” 2020.

Rossi, Mark A., et al. “Prefrontal Cortical Mechanisms Underlying Delayed Alternation in Mice.” *Journal of Neurophysiology*, vol. 108, no. 4, 2012, pp. 1211–1222., doi:10.1152/jn.01060.2011.

Rusin, K. I., et al. “κ-Opioid Receptor Activation Modulates Ca2+ Currents and Secretion in Isolated Neuroendocrine Nerve Terminals.” *The Journal of Neuroscience*, vol. 17, no. 17, 1997, pp. 6565–6574., doi:10.1523/jneurosci.17-17-06565.1997.

Schattauer, Selena S., et al. “Nalfurafine Is a G-Protein Biased Agonist Having Significantly Greater Bias at the Human than Rodent Form of the Kappa Opioid Receptor.” *Cellular Signalling*, vol. 32, 2017, pp. 59–65., doi:10.1016/j.cellsig.2017.01.016.

Schattauer, Selena S., et al. “Peroxiredoxin 6 Mediates Gαi Protein-Coupled Receptor Inactivation by CJun Kinase.” *Nature Communications*, vol. 8, no. 1, 2017, doi:10.1038/s41467-017-00791-2.

Schattauer, Selena S., et al. “Reactive Oxygen Species (ROS) Generation Is Stimulated by κ Opioid Receptor Activation through Phosphorylated c-Jun N-Terminal Kinase and Inhibited by p38 Mitogen-Activated Protein Kinase (MAPK) Activation.” *Journal of Biological Chemistry*, vol. 294, no. 45, 2019, pp. 16884–16896., doi:10.1074/jbc.ra119.009592.

Schmid, Cullen L., et al. “Bias Factor and Therapeutic Window Correlate to Predict Safer Opioid Analgesics.” *Cell*, vol. 171, no. 5, 2017, doi:10.1016/j.cell.2017.10.035.

Shaham, Yavin, and Jane Stewart. “Exposure to Mild Stress Enhances the Reinforcing Efficacy of Intravenous Heroin Self-Administration in Rats.” *Psychopharmacology*, vol. 114, no. 3, 1994, pp. 523–527., doi:10.1007/bf02249346.

Shippenberg, T.s., et al. “Dynorphin and the Pathophysiology of Drug Addiction.” *Pharmacology & Therapeutics*, vol. 116, no. 2, 2007, pp. 306–321., doi:10.1016/j.pharmthera.2007.06.011.

Strang, John, et al. “Opioid Use Disorder.” *Nature Reviews Disease Primers*, vol. 6, no. 1, 2020, doi:10.1038/s41572-019-0144-6.

Swanson, L.w. “The Projections of the Ventral Tegmental Area and Adjacent Regions: A Combined Fluorescent Retrograde Tracer and Immunofluorescence Study in the Rat.” *Brain Research Bulletin*, vol. 9, no. 1-6, 1982, pp. 321–353., doi:10.1016/0361-9230(82)90145-9.

Tcherpakov, Marianna. “Medical Marijuana and the Opioid Crisis.” *BCC Research*, July 2019.

Tejeda, Hugo A, et al. “Prefrontal Cortical Kappa Opioid Receptors Attenuate Responses to Amygdala Inputs.” *Neuropsychopharmacology*, vol. 40, no. 13, 2015, pp. 2856–2864., doi:10.1038/npp.2015.138.

Tejeda, Hugo A, et al. “Prefrontal Cortical Kappa-Opioid Receptor Modulation of Local Neurotransmission and Conditioned Place Aversion.” *Neuropsychopharmacology*, vol. 38, no. 9, 2013, pp. 1770–1779., doi:10.1038/npp.2013.76.

The, T H, and T E Feltkamp. “Conjugation of Fluorescein Isothiocyanate to Antibodies I. Experiments on the Conditions of Conjugation.” *Immunology*, vol. 18, no. 6, 1970, pp. 865–873.

Topell, Simon, et al. “Circularly Permuted Variants of the Green Fluorescent Protein.” *FEBS Letters*, vol. 457, no. 2, 1999, pp. 283–289., doi:10.1016/s0014-5793(99)01044-3.

Tsien, Roger Y. “The Green Fluorescent Protein.” *Annual Review of Biochemistry*, vol. 67, no. 1, 1998, pp. 509–544., doi:10.1146/annurev.biochem.67.1.509.

Veer, Ashlee Van’T, and William A. Carlezon. “Role of Kappa-Opioid Receptors in Stress and Anxiety-Related Behavior.” *Psychopharmacology*, vol. 229, no. 3, 2013, pp. 435–452., doi:10.1007/s00213-013-3195-5.

Vinkenborg, Jan L, et al. “Genetically Encoded FRET Sensors to Monitor Intracellular Zn2+ Homeostasis.” *Nature Methods*, vol. 6, no. 10, 2009, pp. 737–740., doi:10.1038/nmeth.1368.

Volkow, Nora D., and Ting-Kai Li. “Drug Addiction: the Neurobiology of Behaviour Gone Awry.” *Nature Reviews Neuroscience*, vol. 5, no. 12, 2004, pp. 963–970., doi:10.1038/nrn1539.

Volkow, Nora D., et al. “Neurobiologic Advances from the Brain Disease Model of Addiction.” *New England Journal of Medicine*, vol. 374, no. 4, 2016, pp. 363–371., doi:10.1056/nejmra1511480.

Wang, Qi, et al. “Structural Basis for Calcium Sensing by GCaMP2.” *Structure*, vol. 16, no. 12, 2008, pp. 1817–1827., doi:10.1016/j.str.2008.10.008.

Watkins, Linda R., et al. “Glial Activation: a Driving Force for Pathological Pain.” *Trends in Neurosciences*, vol. 24, no. 8, 2001, pp. 450–455., doi:10.1016/s0166-2236(00)01854-3.

Wettschureck, Nina, and Stefan Offermanns. “Mammalian G Proteins and Their Cell Type Specific Functions.” *Physiological Reviews*, vol. 85, no. 4, 2005, pp. 1159–1204., doi:10.1152/physrev.00003.2005.

White, Kate L., et al. “Identification of Novel Functionally Selective κ-Opioid Receptor Scaffolds.” *Molecular Pharmacology*, vol. 85, no. 1, 2013, pp. 83–90., doi:10.1124/mol.113.089649.

Wikler, Abraham. “Conditioning Factors in Opiate Addiction and Relapse.” *Journal of Substance Abuse Treatment*, vol. 1, no. 4, 1984, pp. 279–285., doi:10.1016/0740-5472(84)90008-4.

Wise, Roy A. “Dopamine and Reward: The Anhedonia Hypothesis 30 Years On.” *Neurotoxicity Research*, vol. 14, no. 2-3, 2008, pp. 169–183., doi:10.1007/bf03033808.

Woolley, J.f., et al. “Recent Advances in Reactive Oxygen Species Measurement in Biological Systems.” *Trends in Biochemical Sciences*, vol. 38, no. 11, 2013, pp. 556–565., doi:10.1016/j.tibs.2013.08.009.

Wright, Patrick W., et al. “Functional Connectivity Structure of Cortical Calcium Dynamics in Anesthetized and Awake Mice.” *Plos One*, vol. 12, no. 10, 2017, doi:10.1371/journal.pone.0185759.

Xu, M., et al. “Sciatic Nerve Ligation-Induced Proliferation of Spinal Cord Astrocytes Is Mediated by Opioid Activation of p38 Mitogen-Activated Protein Kinase.” *Journal of Neuroscience*, vol. 27, no. 10, 2007, pp. 2570–2581., doi:10.1523/jneurosci.3728-06.2007.

Yasuda, R., et al. “Imaging Calcium Concentration Dynamics in Small Neuronal Compartments.” *Science Signaling*, vol. 2004, no. 219, 2004, doi:10.1126/stke.2192004pl5.

Yin, Henry. “The Role of the Murine Motor Cortex in Action Duration and Order.” *Frontiers in Integrative Neuroscience*, vol. 3, 2009, doi:10.3389/neuro.07.023.2009.

Zan, Gui-Ying, et al. “Antagonism of κ Opioid Receptor in the Nucleus Accumbens Prevents the Depressive-like Behaviors Following Prolonged Morphine Abstinence.” *Behavioural Brain Research*, vol. 291, 2015, pp. 334–341., doi:10.1016/j.bbr.2015.05.053.

7 Appendices

7.1 Appendix I: Capstone Thesis

Developing a Tool for Analyzing Continuous Self-Administration Patterns of Opioids in Rodent Models

Brenden Wong

Chavkin Lab

June 7, 2019

Abstract

Morphine and other opiate addictions are now considered an epidemic in the United States. Therefore, pre-clinical research is increasingly directed to finding treatments for individuals suffering from substance use disorders. Our preliminary studies showed that drug self-administration patterns in rodents can be examined through measurement of opioid gelatin consumption. However, the mass of remaining gel could only be measured by an experimenter intermittently, resulting in an incomplete profile of morphine gel consumption. Therefore, we needed a method capable of continuously measuring the mass of gel left in the cage. To this end, we created a device that uses the change in electrical output due to the physical deformation of strain gauges in a load cell to determine and record the mass of gel left in the cage at any given time. Determining the consumption profile of morphine, cannabinol (CBD), and Δ9-tetrahydrocannabinol (THC) gels in comparison to control (drug-free) gels will elucidate the effect of morphine consumption on mouse behavior. The device also monitored desiccation rates of the gel and any potential confounding effects on the data by continuously measuring the mass of gel in the device in an empty cage throughout the day. This device will allow for a richer analysis of self-administration of drugs throughout different times of the day and, in the long term, will aid in the development of more effective therapeutics for opiate addiction in humans.

Contents

Introduction

In recent years, opioids have become increasingly problematic as addictive substances, being branded as “the opioid crisis”. This is partially due to the large number of fatalities and economic burden placed on society as a whole. For example, over 115 people die in the United States alone annually and the United States spends around $78 billion on dealing with opioid addictions[1,2]. Morphine, heroin, and fentanyl are all examples of these types of opioids that cause these types of addictions. In addition, addictions to oral opioids can lead into more dangerous drug abuse, such as intravenous heroin use. These addictions are especially dangerous because prolonged use of the substance often results in tolerance and escalation of dosing[3]. This crisis could be addressed potentially be tightening restrictions on opiate-based prescription painkillers. However, we were more interested in developing better pharmacological treatments for individuals as opposed to such regulations. It has recently become the focus of pharmacological research to find an alternative painkiller or analgesic to prevent these addictive behaviors from occurring in the first place.

To aid in this research, animal models have been developed to simulate and model patterns of self-administration of opioids and the effect of this consumption on behavior. Specifically, mouse models have proven to be the most useful and economically feasible in modeling addiction circuits in humans. As is the case with humans, increased stress or pain often increase the consumption and self-administration patterns of these types of drugs. Using the mouse model to study how these addiction circuits work, a significant amount of research has been performed on the morphine/mu-opioid receptor (MOR) system to identify key components in signaling pathways activated by the MOR. In recent studies, it has been identified that MOR activation results in activation of the β-arrestin pathway in addition to G-protein pathways. β-arrestin pathways are often associated with respiratory depression and the G-protein pathways are often associated with analgesia[4]. Therefore, by finding a way to disentangle these two pathways, where we can activate the G-protein pathway without the β-arrestin pathway, we could develop a more effective and non-addictive therapeutic. By analyzing gel consumption patterns, it allows for a more in-depth analysis of how various opioids and other drugs lead to addictions and could aid in elucidating the differences between the two pathways.

Biomedical Need

While it has been determined that the G-protein and β-arrestin pathways are triggered by MOR activation, it remains unclear to what extent each of these pathways are affected by other physiological mechanisms and how these pathways interact with each other as well as other signaling pathways. Eventually, we would like to understand the voluntary self-administration of morphine for humans and this study helped to clarify the behavioral effects caused by morphine in mice. We expected that once the mice became addicted, there would be an increased amount of morphine consumed compared with the control gels, which was supported by the experimental data. A gel self-administration protocol had already been established, which was used in this study.

Therefore, knowing the mass of gel consumed and the feeding pattern in which that gel was consumed was critical for understanding how morphine self-administration affects mouse behavior. For example, we could gain insight into how the drug consumption affects sleep behavior of the mice and if mice with escalated consumption show significant activity during the dark cycle. However, the current experimental procedure only allowed for the analysis of the total amount of gel consumed and not the feeding pattern.

In this study, we designed a device for measuring and recording the mass of gel left in the cage continuously throughout the course of the day. It needed to be sensitive enough to measure small fluctuations in mass, be resistant to damage inflicted by the mice, and capable of recording data to a file to be analyzed later. In addition, we created MATLAB code capable of analyzing and extracting useful behavioral information from the data.

Current Technology

Mouse models have been frequently used by researchers to investigate certain addiction circuits and have been used in many other studies. Because of this, there is a wealth of existing technologies for studying the effect of addictive drugs on mice. Many of these technologies use methodologies different from the gel self-administration protocol.

In one study, a lickometer was used to measure ethanol drinking behavior in mice[5]. While this study does not deal with addictive drugs in a gelatin media, it still addresses the drug self-administration patterns in mice.

Another methodology that investigates the effect of addictive drugs on mice behavior involves the use of intraperitoneal injections[6]. These types of studies involve the investigation of addictive drugs on behavior, but don't address the self-administration of drugs, which don't necessarily accurately represent reality in human models.

Finally, other studies involve the use of a lever-pressing mechanism to allow for intravenous self-administration[7]. As opposed to the daily intraperitoneal injections, the lever-pressing mechanism allows for continuous self-administration. One issue with this would be the mechanism of delivery and considerations necessary when designing the device to prevent the mice from damaging it.

However, these studies all fall short of allowing for a convenient methodology for recording and analyzing the self-administration patterns and effects on behavior since they lack the capability of continuously measuring and recording the amount of gel consumed by the mice. Therefore, we designed a device to be used in the mouse cages that allowed for this data collection and analysis to extrapolate the effect of morphine and other drugs on behavior, especially after events modeling chronic pain occurred in the mice.

In designing such a device, it is important to take into consideration various aspects and constraints and needs for the device to be successful, as mentioned previously. Some of the additional important considerations that were taken into account were the safety of the mice, the social and ethical considerations for working with mice, and the economic factors for designing the scales.

Design Solution

The following sections will describe the various iterations performed on the design solution.

Design Specifications

When designing the device, we needed to consider key aspects of how it should function and what it should be able to do. We wanted the device to be capable of accurate and continuous monitoring of the mass of gel left in the cage, be resistant to tampering from the mouse, be sterilizable, and be able to function while on the mouse cage rack.

In designing the solution, we also needed to consider limitations imposed by using an Arduino Uno for data acquisition. There are only a certain number of pins available for the load cell to use, so we were limited in the capabilities that we wanted. Initially, it was proposed to use an IR beam breaker system to determine mouse locomotion and confirm eating events. However, we decided that it would produce an excessive amount of strain on the Arduino and that the data gained from that would ultimately be redundant. Therefore, it was not included in the final design solution.

In addition, the load cell needed to be secured in a fashion that allowed for the proper bar deformation to occur, but that also prevented the mouse from destroying the wiring on the load cell. In the initial design iterations, the underside of the load cell was unprotected, which allowed the mouse to chew on and destroy the wiring, which is shown in Figure 1.

Engineering Standards

An integral part of the device design was the 3D printed components that allowed for compatibility with the mouse cage and the rest of the device design. The ISO Standard 527-1:2012 applies in this study to characterize the general tensile properties of thermoplastic polyesters in general[8]. In this specific case, the thermoplastic polyesters that were used were polylactic acid (PLA) and polyethylene terephthalate glycol (PETG).

Figure 1: *Example of mouse destroying load cell due to not protecting the load cell wiring. A design iteration was performed on the base plate bottom to account for this and to more fully protect the load cell wiring.*

Animal Regulatory Standards

Since this study involves the use of animal test subjects, it was necessary to receive approval from regulatory bodies such as the Institutional Animal Care and Use Committee (IACUC), and the National Institute of Health (NIH). The treatment of mice test subjects was followed according to the 1985 regulation enacted by the NIH entitled the “Public Health Service Policy on Humane Care and Use of Laboratory Animals”[9]. In addition, protocol involving the specific treatment of animals was submitted and approved by the

IACUC under the Chavkin Lab Protocol number 2153-08, as specified by the publication for this study[10].

Considerations

In addition to the previously mentioned technological, regulatory, and design constraints, other considerations were also made regarding various limiting aspects. In this case, certain aspects such as public health considerations, global considerations, cultural considerations, and social considerations are all not applicable to this design solution. However, we needed to consider the safety and welfare of the mice with which we performed the drug self-administration and pain tests. The tests needed to be performed in an approved and ethical manner. In addition, there were environmental considerations that ensured that the device served as a safe environment for the mice and the device needed to be relatively cheap to manufacture, since we created eight separate devices.

In addition, some other constraints that were taken into consideration were the constructability, maintainability, and usability. The device needed to be relatively easy to construct, which would allow for ease of use by the experimenter. In addition, it needed to be reproducible by other experimenters. To this end, an instruction manual was created to guide other users on how to construct and operate the device. Finally, it was important that the device was easy to use, since this would affect the utility as a tool for future experiments.

Based on the all the previous considerations, Table 1 summarizes the specifications that were considered when designing the device.

Table 1: Design Specifications

Need #	Stakeholder	Need	Rank
1	Brenden	The device is resistant to damage by mice	1
2	Brenden	The device continuously monitors gel mass	1
3	Brenden	The device is capable of measuring small mass fluctuations	1
4	Brenden	The device is sterilizable	3
5	Brenden	The device has no influence on frequency of eating events	2
6	Brenden	The device does not need to be connected to a computer to operate	2
7	Brenden	The device is easy to operate	2
8	Brenden	The device fits within a single mouse cage	1
9	Brenden	The device never goes below a minimum height above the floor of the cage	2
10	Brenden	The device is compatible with the mouse cage	1
11	Brenden	The device saves data in a manner that is able to be analyzed separately	1
12	Brenden	The device is safe for mice to interact with	1
13	Brenden	The device is easy to construct	3
14	Future Experimenters	The device is reproducible by other experimenters	2
15	Dr. Chavkin	The device is economically feasible	3

First Design Iteration

The initial solution that was used consisted of a load cell, a load cell amplifier, an Arduino for data acquisition, various 3D printed pieces, and analysis of the data with MATLAB. An SD card module and LED display were added later to allow for easier use of the device. The general flow diagram for how the solution works is shown in Figure 2.

The design works by first loading the gel cup with the prepared gelatin. Once the gel is loaded, the weight causes a deformation in the load cell, which deforms the strain gauges

located on the top and bottom of the cell. This deformation results in a change in electrical output, and this electrical signal is sent to the load cell amplifier, and then passed to the Arduino. The Arduino writes the measurement to an SD card, which can then be extracted later and analyzed using MATLAB.

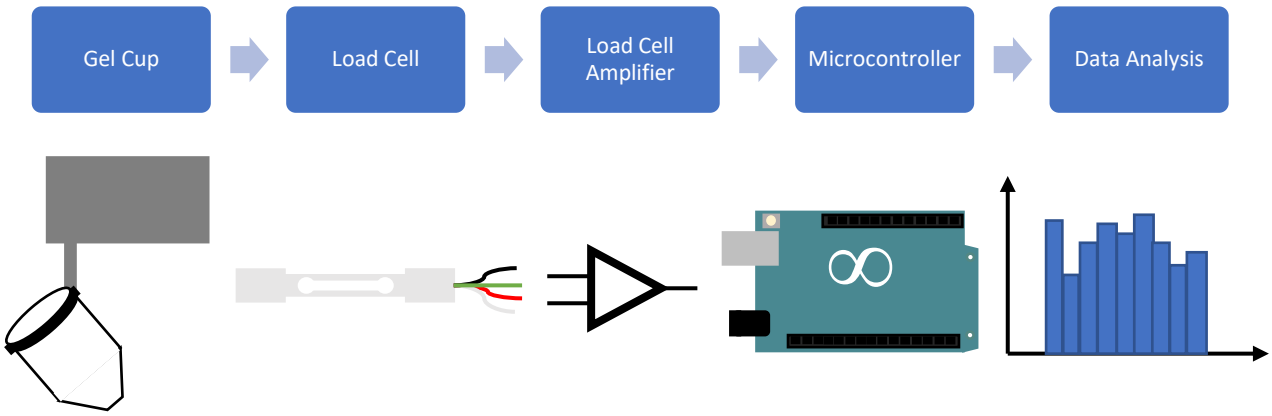


Figure 2: *Flow diagram for the design solution. The gel cup is suspended from the load cell, and the mass of gel in the cup causes a deformation in the strain gauges in the load cell. The electrical signal is amplified through a load cell amplifier and passed to the Arduino. The data is then written to an SD card and saved. for processing in MATLAB.*

To aid in processing the data, we created a MATLAB script capable of importing the data and separating the eating events from the mass of gel remaining in the cup. In addition, the script smooths the mass data to remove any artifact caused by the electronics of the circuit. The script then writes the separated data to a separate file. This is a novel solution because it provides a method for monitoring mouse behavior from drug self-administration using an automated device. By designing such a device, it allowed for research to be effectively performed to allow for richer sets of data to help inform the accuracy of our hypothesis.

Initially, our hypothesis regarding the experimental results was that mice would consume gel with morphine and thereby self-administer morphine. We expected that feeding patterns would show the developed drug dependence of the mice and that consumption would differ throughout the course of the trials as the mice would develop a tolerance to the analgesic effects of morphine. Finally, we expected that mice would consume higher levels of morphine while in a state of chronic pain, serving as a model for individuals suffering from this type of chronic pain.

Second Design Iteration

After various stages of troubleshooting, an LED display was added to the circuitry to allow for monitoring the data while the scale was running. Previously, there was no method for knowing what mass the scale was reading and there was significant difficulty experienced with troubleshooting. By adding the LED display, it made troubleshooting easier and more feasible. In addition, seven more devices were created to allow for tandem use for a cohort

of eight mice. Besides these iterations, the 3D printed pieces were constantly modified to allow for optimum electrical isolation and efficiency of design.

Materials and Methods

The following sections will describe the main methods for designing and testing the device and for the protocol used to administer the gels to the mice. In addition, the statistical methods for analyzing the data will be included. For initial prototyping, we created a single scale. After troubleshooting and finalizing a design, we created seven more scales to be used in tandem for a total of eight mice in an experimental cohort. Design for a single scale will be explained in the following section. The gel self-administration protocols were followed as set by Leung et al.[10]

Gel Self-Administration Protocol

In the study, morphine, Δ9-tetrahydrocannabinol (THC), and cannabidiol (CBD) were added to gelatin for mouse consumption. THC and CBD were included to compare the analgesic effects with those of morphine. The gelatin was prepared initially by adding 3.85 g of Knox Gelatin in 100 mL of deionized water, with 5 g (5 % sugar content) of polycal sugar to acclimate the mice to eating the gels. After two to three days, the gels were prepared with 2.5% sugar content and administered to the mice for two to three days, at which point the gel was prepared with 1% sugar content and administered to the mice. Once the mice were fully acclimated to the gels, the drug in question was added to reach a certain concentration (1 mg/ 15 ml for THC and CBD, 1.125 mg/15 ml for morphine) was infused in 1% sugar content gel and placed in the cage for self-administration. The mass of gel was weighed prior to placement in the cage and was measured again 24 hours later to determine the total amount of gel consumed by the mouse.

Design Methods

The following sections will describe the methods used for designing key components for cage compatibility as well as methods used for data acquisition and for data analysis.

Computer Aided Design

One of the key components of designing the device was using computer aided design. The various components that were crucial to the compatibility of the mouse cage were designed using Autodesk Professional Inventor 2019. The four pieces that were created were a base plate, a bottom piece for the base plate, a gel cup holder, and a gel cup lid, which are shown hereafter in Figure 4. The partial assembly for these pieces is shown in Figure 3.

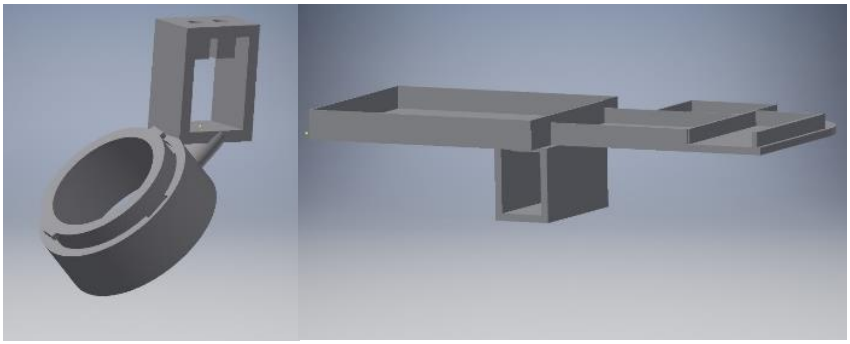


Figure 3: *Assembly of the gel cup holder with the gel cup holder top, with the top rotated to locking position (left). The base plate bottom attached to the bottom of the base plate (right). The load cell was inserted into the base plate bottom and the other end of the load cell was connected to the holes in the top of the gel cup holder.*

The base plate was designed to allow isolation from the mice with the circuit components and to protect the circuit from damage as well as to protect the mice from injury. It was designed to rest on top of the metal mouse cage and to contain the Arduino, the breadboard, and the load cell amplifier.

The base plate bottom was designed to hold the load cell. As mentioned previously, the base plate bottom provided protection against damage by the mice and also allowed for the load cell to be suspended from the top of the mouse cage. The base plate bottom was designed with holes to be secured to the bottom of the base plate.

The gel cup holder was designed to be attached to the opposite end of the load cell and be suspended from the top of the mouse cage. The gel cup was placed inside the ring of the gel cup holder and provided a secure attachment that allowed the mice to freely eat from the gel cup.

The gel cup lid was designed to sit on the lip of the gel cup to adapt and fit inside the gel cup holder. It was designed to lock in place with the ring in the gel cup holder and prevent the mice from dislodging the gel cup.

Prototype Testing

Once the device was created, it was tested initially computer powered and was loaded with a known weight. The data was written to a file on the computer for a period of around 19 hours. This testing was performed to ensure that the device was functional and was capable of correctly recording a known weight.

After it was confirmed that the device was working, a mass of gel was placed in the gel cup and left for a period of around 24 hours. This testing was performed to measure the rate of gel desiccation in an empty cage, which was important to consider if the reduction in mass was due largely to mouse gel consumption in experimental trials or gel desiccation. In this phase, the device was still computer powered.

After the initial testing to ensure that the device was capable of accurately measuring mass for an extended period of time, the experimental trials was performed using a single mouse

over a period of around 18 hours. In this run, the eating events were recorded and a were represented as a threshold value of 20 g.

To enable the device to be used in the mouse cage racks, we incorporated the capability to write data to a microSD card, which allowed the device to be powered by an outlet instead of a computer. This allowed for more flexibility in running the experiments. Using this setup, the same experiment was performed with a mouse and control gel consumption.

Finally, once the device was confirmed to be fully functionally in characterizing mouse self-administration patterns, we created seven more devices to be used in tandem for a total of eight mice in a single cohort. This allowed for a maximum capacity of two mice for each drug group.

Experimental Methods

The trials were performed using the previously mentioned gel self-administration protocol for multiple days with a cohort of eight mice and were repeated with a cohort of six mice. The data were subsequently recorded and analyzed.

For the first cohort of eight mice, the consumption patterns of the mice were collected every day for 7 days. There was a total of four groups: control, THC, morphine, and CBD, and each group had two mice.

For the second cohort of six mice, the consumption patterns of the mice were collected every day for 8 days. There were two mice in the control and morphine groups and one mouse in each of the THC and CBD groups.

Pain Tests

To test the escalation of drug consumption, we measured the initial mass of gel consumed by the mice and we measured their response to pain tests such as a hot plate test, a Von Frey test, and a Hargreaves test, which are described hereafter. A partial sciatic nerve ligation (pSNL) was then performed on the mice and we then measured their response to the same pain tests. We also observed the differences in mass of gel consumed by the mice.

Hot Plate Test

The hot plate test of response latencies measures nociception, and an increase in sensitivity is typical of the hyperalgesia experienced by the mice after the partial sciatic nerve ligation. The mice were placed on a hotplate at 57.5°C and were tested for the latency to lick their paws or jump.

Von Frey Test

The Von Frey test measured the allodynia experienced by the mice after the partial sciatic nerve ligation and was based off previous experimental trials testing the allodynia. The

test was performed by placing the mice on a metal mesh and the surface of the mouse paws was touched with different Von Frey filaments with bending forces from 0.166 g to 3.63 g and the time it takes for paw withdrawal was measured.

Hargreaves Test

The Hargreaves test measures the hyperalgesia experienced by the mice by directing a high-intensity beam of light at the hindpaw. The mice were tested for the latency to withdraw or lick its paw. The results were recorded and used as a measure for the increased sensitivity.

Statistical Analysis

The accuracy of the scale was classified using a simple linear regression model with a correlation coefficient. After multiple trial runs for each drug, the data were compiled and processed. The data for each drug was averaged and a simple standard error was generated for each point, which was shaded as Mean±SE. In addition, the number of eating events were classified into bouts based on the time between eating events and were counted for each day. These were then analyzed via ANOVA to determine whether the number of bouts were statistically different between each of the drug groups.

Results

The following sections will describe the results from the device design, from the testing of the design iterations, and the experimental results regarding the gel self-administration of drugs.

Device Design: First Iteration

For the design of the device, we used the following materials: an Arduino Uno, an HX711 Load Cell Amplifier, a 100 g Load Cell, jumper wires, one mini breadboard, one microSD card reader module, one microSD card, one USB printer cable, eight M3 12mm machine screws, and one gel cup. In addition, we designed various pieces for 3D printed to allow for compatibility with the mouse cages. The pieces were designed using Autodesk Professional Inventor 2019 and 3D printed using polyethylene terephthalate glycol (PETG). These pieces include a base plate, a base plate bottom, a gel cup holder, and a gel cup lid, which are shown and described in Figure 4.

Some of the individual components also needed some preparation prior to assembly. The load cell amplifier required the soldering of breakaway header pins. 90-degree header pins were used in some of the scales to facilitate wiring but were not necessary.

To allow for compatibility with the rest of the system, the load cell required some additional wire soldering. Superglue was used at the base of the load cell on the wiring to increase the strength, since that area was the most fragile and prone to damage or breaking. Two female-female jumper cables were cut in half and attached to the ends of the load cell

leads. These connections were then soldered and shrink wrapped to connect to the pins on the load cell amplifier.

To prepare the gel cup itself, the gel cup was fitted inside the gel cup lid. Once the gel cup was placed in the lid, it could be placed in the holder and rotated 90 degrees to lock it in place. It was still able to rotate but was not removable from the gel cup. To remove it, it could be twisted until the tabs of the gel cup holder lid piece were aligned with the notches in the gel cup holder.

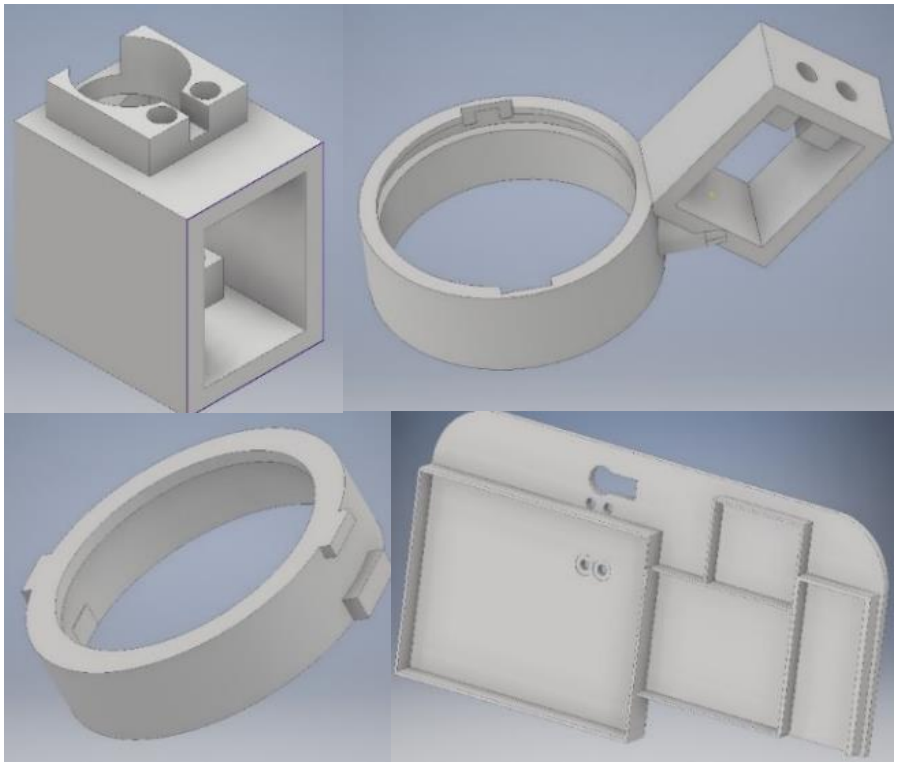
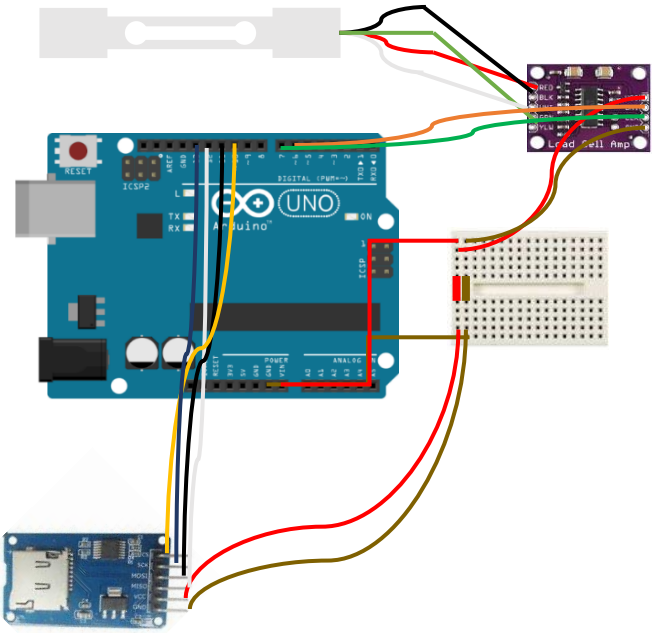


Figure 4: *CAD models of 3D printed scale components. (Top Left) The base plate bottom houses the load cell and protects the wiring from damage by the mice. (Top Right) The gel cup holder allows for quick and easy removal of the gel cup for loading gel into the cage. (Bottom Left) The gel cup lid attaches to the lip of the gel cup to allow for compatibility with the gel cup holder. (Bottom Right) The base plate provides isolation of the circuit components and allows for placement of the various circuit components.*

To configure the circuit, the diagram shown in Figure 5 was used.

The load cell was connected to the load cell amplifier, which was in turn connected to the corresponding pins in the Arduino. The data from the load cell was passed through the amplifier to the Arduino and written to the microSD card via the microSD card reader module. The microSD card could then be removed to analyze the data. While the Arduino could write directly to the computer, a microSD card was used to allow for multiple Arduinos to be used simultaneously.



Arduino to Load Cell Connections	
Amplifier	**Arduino**
VCC	Vcc
DAT	Pin 6
CLK	Pin 7
GND	Gnd
Amplifier to Load Cell Connections	
Amplifier	**Load Cell**
RED	Red wire
BLK	Black wire
WHT	White wire
GRN	Green wire
SD Card Module to Arduino Connections	
SD Card Module	**Arduino**
CS	Pin 10
SCK	Pin 13
MOSI	Pin 11
MISO	Pin 12
VCC	Vcc
GND	Gnd

Figure 5: *(Left) Wiring diagram to show the connections between the load cell, the load cell amplifier, the Arduino, and the microSD card module. (Right) Description of connections between load cell, load cell amplifier, Arduino, and microSD card module*

Device Assembly

To assemble the scale, the baseplate was first placed on the wire cage top. The wired end of the load cell was inserted in the base plate bottom and the holes of the load cell were aligned with the holes on the bottom of the base plate bottom piece. Screws were inserted through the base plate bottom holes and the load cell holes to ensure that the load cell remained level. The wires were fed from the load cell through the large hole in the top of the base plate bottom. The other end of the load cell was placed underneath the holes from the gel cup holder and the gel cup holder holes were aligned with the load cell holes and were secured with screws. From underneath the wire cage top, the four holes and the large hole with the wires from the base plate bottom piece were aligned with the four holes and the large hole in the base plate. The wires from the load cell were fed through the large hole of the base plate and the base plate bottom was secured with screws to ensure that the base plate bottom remained level. The final device assembly is shown in Figure 7.

Device Operation

The scale was first calibrated using the Load_Cell_Calibration sketch and uploaded to the Arduino, which is included in Appendix A and was created by Nathan Seidle[11]. Once a known mass was placed in the gel cup, we used the serial monitor on the Arduino IDE to

adjust the calibration factor until the displayed value on the serial monitor matches the actual mass. Once this calibration factor was known, it was included in the Load_Cell_Measurement sketch and uploaded to the Arduino, which is also included in Appendix A.

The scale was then placed in the mouse cage and left running for a period between 18 to 24 hours. A mass measurement was recorded on the microSD card every 2 seconds. After extracting the microSD card, the file was renamed to incorporate the mouse number and the date and stored for data processing.

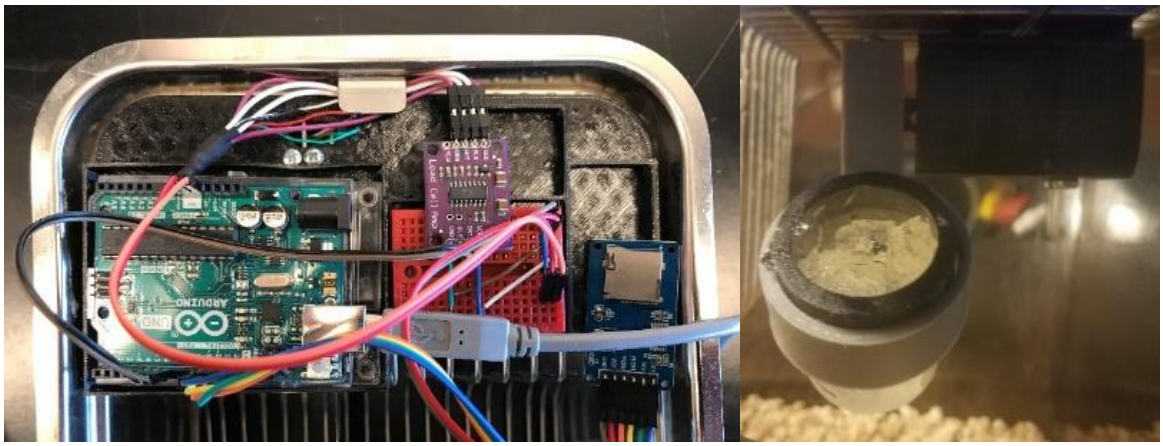


Figure 6: *Final setup for the initial prototype of the device. Circuit components consisting of the Arduino, load cell amplifier, and breadboard connected as described (left). Base plate bottom with load cell attached and gel cup holder attached to the load cell, with the gel cup loaded into the gel cup holder (right).*

Data Processing

The files were processed using the DataProcessing MATLAB script, which is included in Appendix B. The script prompts the user for the mouse number and date that the experiment was performed. This reads the data file in and extracts the data. Disturbances were classified as such if there was a difference greater than 0.4 g between two consecutive samples, which were separated by 2 seconds. This threshold was applied to the data such that disturbances caused by the mice were maxed to an arbitrary value. The data was then separated into distinct eating events and mass of gel left in the cup. This data was then exported and stored according to mouse number and drug. In addition, the distinct eating events were processed such that any period greater than two minutes between disturbances was considered a simple mouse disturbance and was excluded as an eating event. The number of eating events were then distributed into time bins to determine an eating profile of the mouse, which was then exported as well.

Device Testing

The device was initially run for a period of around 19 hours. In this test, there was a known weight placed in the holder to allow for correct calibration with no mouse to test the

functionality of the device. It was observed that the mass remained relatively constant with very little fluctuations.

The device was then run with a mass of gel in the holder to measure desiccation rates of the gel. After testing the functionality of the device, we then tested the device with a live mouse to verify that the mouse would not damage the device. The results of these runs are shown in Figure 8.

Finally, the device was modified to write the data to a microSD card instead of writing the data to a file on the computer and to plot the mass of gel consumed instead of the mass of gel remaining in the cup. The results of this run are also shown in Figure 8, this time without the distinct eating events.

After running the scales with various mice, the final mass of gel consumed recorded by the load cell was compared with the mass of gel consumed recorded by the user. It was expected that if the scale worked perfectly, the best fit line would be $y = x$ and $R^2 = 1$. The resulting correlation is displayed in Figure 10, with a best fit line of $y = 0.928x + 0.2723$, with $R^2 = 0.9672$.

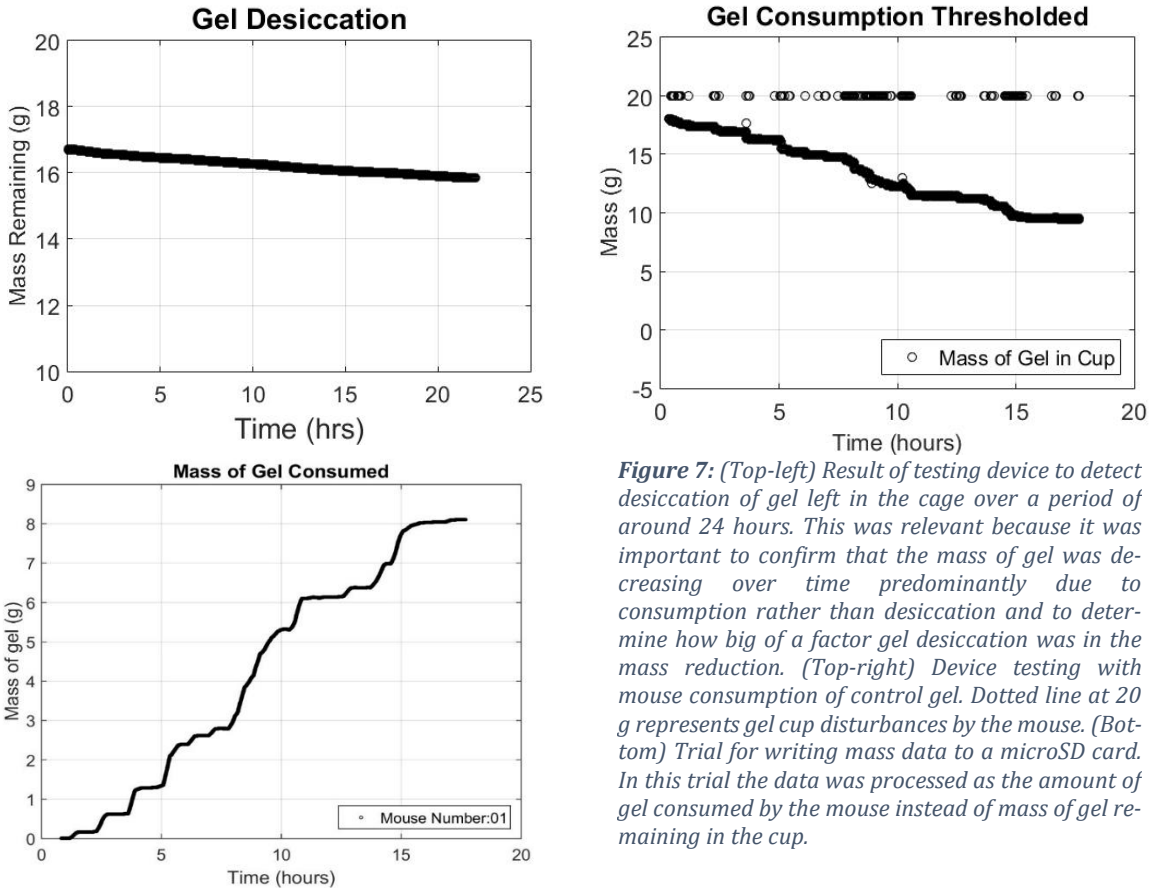


Figure 7: *(Top-left) Result of testing device to detect desiccation of gel left in the cage over a period of around 24 hours. This was relevant because it was important to confirm that the mass of gel was decreasing over time predominantly due to consumption rather than desiccation and to determine how big of a factor gel desiccation was in the mass reduction. (Top-right) Device testing with mouse consumption of control gel. Dotted line at 20 g represents gel cup disturbances by the mouse. (Bottom) Trial for writing mass data to a microSD card. In this trial the data was processed as the amount of gel consumed by the mouse instead of mass of gel remaining in the cup.*

Experimental Results

From the different test groups, we collected consumption data for various mice for multiple days. Table 2 summarizes the drug groups, the number of mice in each, and the total number of days of data collected for each.

Table 2: Experimental Groups

Group	Number of Mice	Days of Data
Control	4	21
THC	3	17
Morphine	4	26
CBD	3	19

For each of the groups, we collected the amount of gel consumed throughout an 18-hour period. Five characteristic days for each of the drug groups are plotted in Figure 11. After all the experimental trials had been performed, the mass of gel remaining in the cup were averaged and plotted for each of the drug groups. In addition to the gel averages, the number of eating bouts for each of the days for each group of drugs was averaged. Both plots are shown in Figure 12.

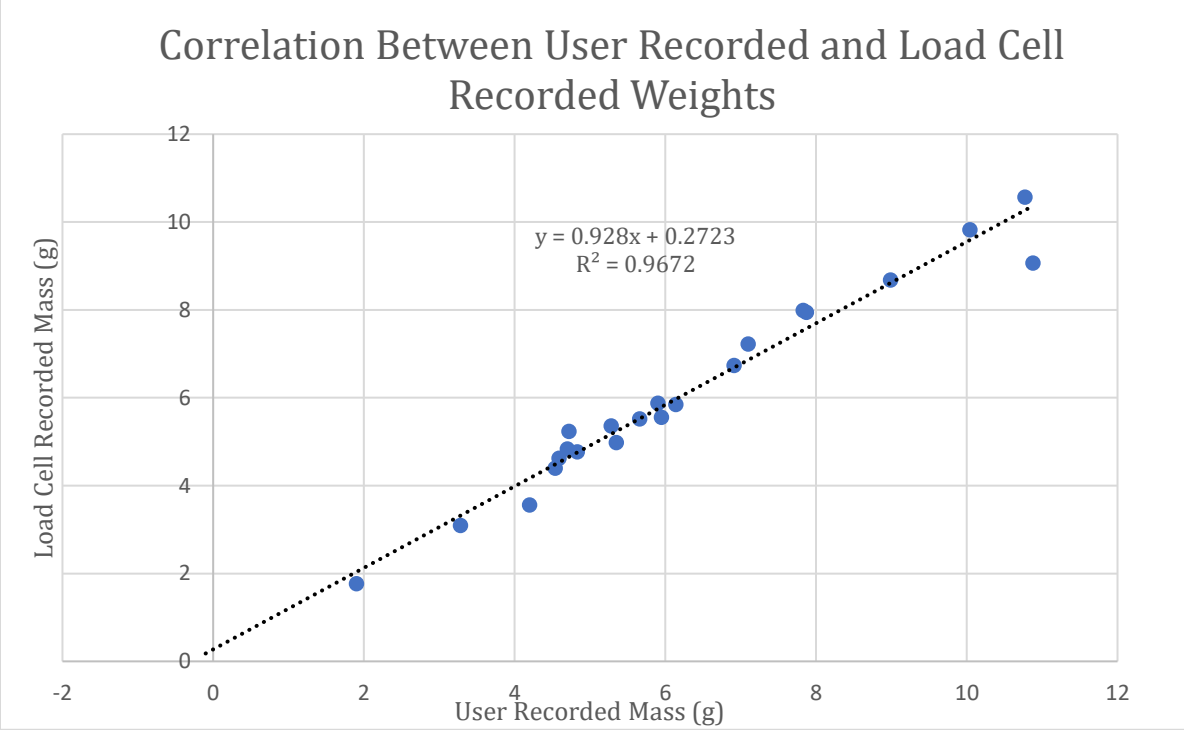


Figure 8: *Correlation between the mass of gel consumed recorded by the user and the mass of gel consumed recorded by the load cell. The correlation was done with data from 6 mice over 4 days, 2 trials of which were unusable due to load cell malfunctioning. Therefore, it was done with an n = 22 days of data.*

Pain Test Data

The pain tests were performed both pre and post partial sciatic nerve ligation (pSNL) and the results from the von Frey test are represented in Figure 13. for all the groups of drugs.

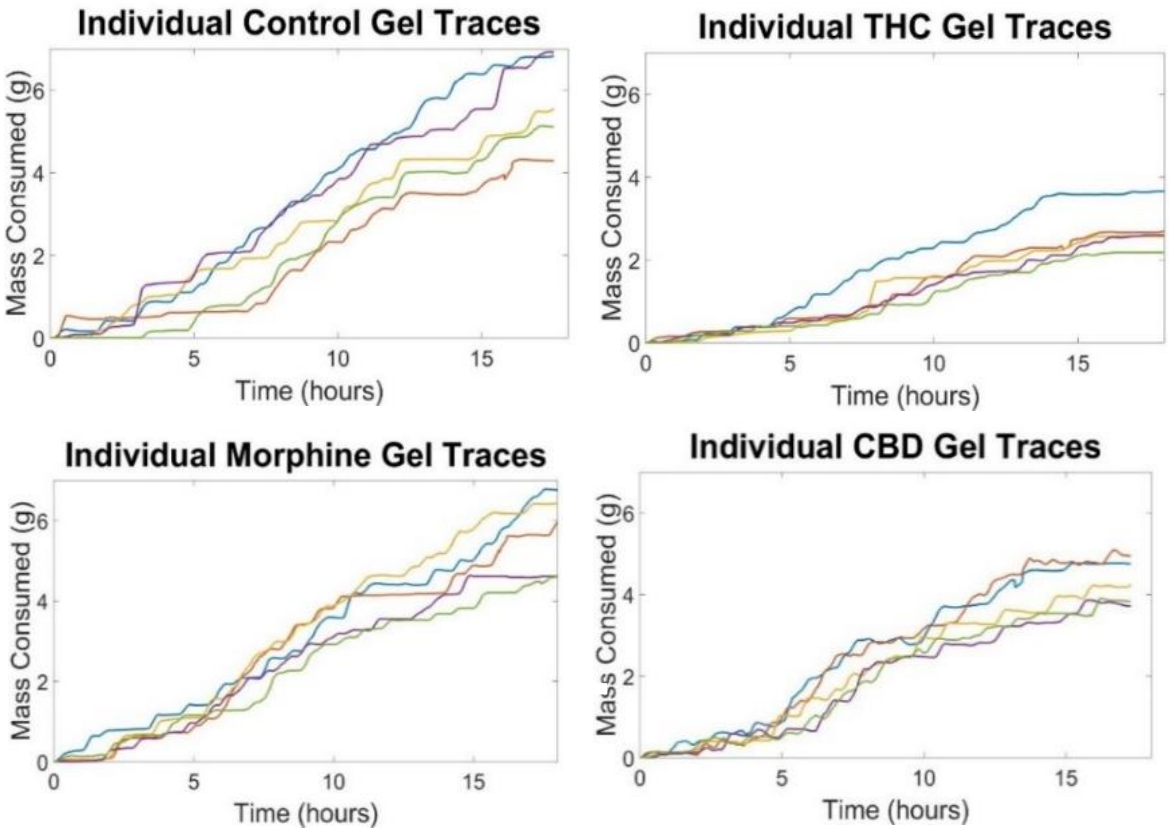


Figure 935: *Characteristic individual traces for five days for each of the drug groups over 16 hours of recording. Individual traces illustrate the feeding microstructure of the mice throughout the course of the day. Mice consumed significantly less THC than the other drugs.*

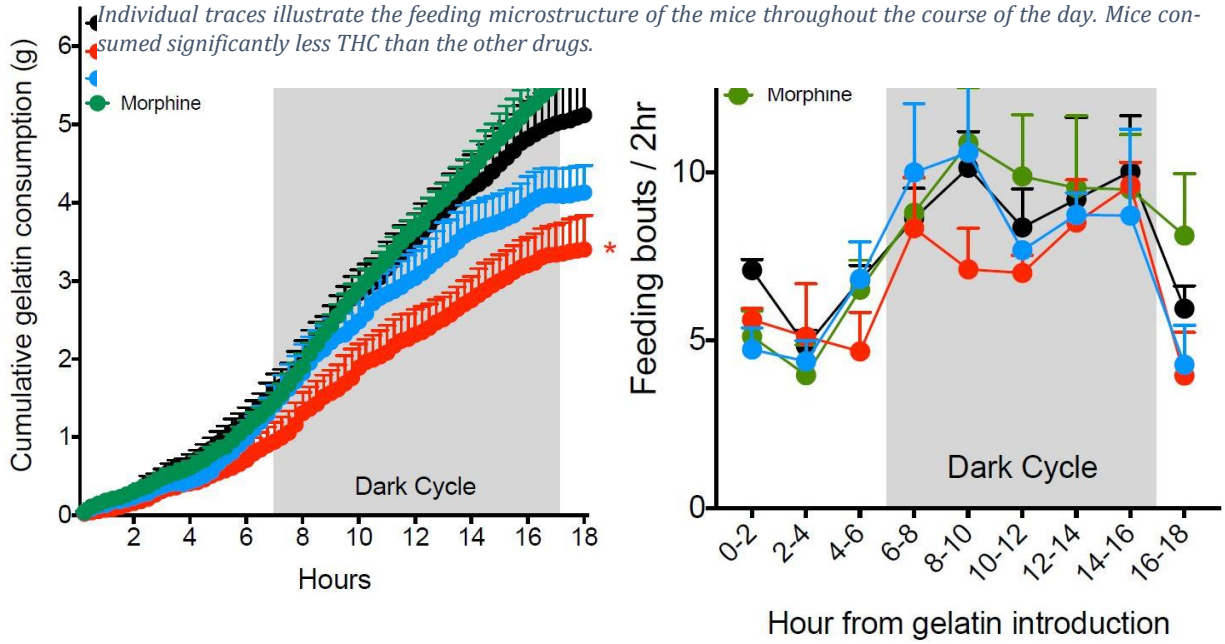


Figure 10: *(Left)) Averages of gel consumption over a course of 18 hours. Morphine gel exhibits the highest mass of gel consumption while THC exhibits the lowest mass of gel consumption. (Right) Distribution of bouts throughout the day for each of the drug groups. Bouts were characterized as such when there was a period greater than two minutes between subsequent disturbances.*[10]

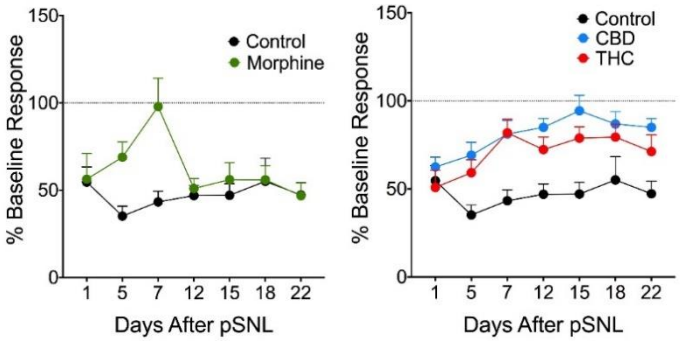


Figure 11: *Results from von Frey test comparing control with morphine response (left) and comparing control with CBD and THC response (right).*

The von Frey test was performed to measure allodynia in the mice. It was observed that directly after pSNL, the morphine response increased almost back to the baseline response, while the control group was reduced to nearly half of the baseline response. However, after a week, the response of the morphine group returned to the reduced control response. In contrast, the CBD and THC responses both show an increase towards the baseline with no consequent reduction.

Discussion

In the following section, the final device design will be analyzed and compared with the initial specifications set forth at the beginning of the project. The experimental results will be analyzed and interpreted in the context of current and future research.

Table 3: Comparison of Final Device with Initial Specifications

Need #	Stakeholder	Need	Rank	Met?
1	Brenden	The device is resistant to damage by mice	1	Yes
2	Brenden	The device continuously monitors gel mass	1	Yes
3	Brenden	The device is capable of measuring small mass fluctuations	1	Yes
4	Brenden	The device is sterilizable	3	Semi
5	Brenden	The device has no influence on frequency of eating events	2	Yes
6	Brenden	The device does not need to be connected to a computer to operate	2	Yes
7	Brenden	The device is easy to operate	2	Yes
8	Brenden	The device fits within a single mouse cage	1	Yes
9	Brenden	The device never goes below a minimum height above the floor of the cage	2	Yes
10	Brenden	The device is compatible with the mouse cage	1	Yes
11	Brenden	The device saves data in a manner that is able to be analyzed separately	1	Yes
12	Brenden	The device is safe for mice to interact with	1	Yes
13	Brenden	The device is easy to construct	3	Semi
14	Future Experimenters	The device is reproducible by other experimenters	2	Yes
15	Dr. Chavkin	The device is economically feasible	3	Yes

Design Analysis

After the device was implemented and used experimentally, it was analyzed to determine how well the device met the original specifications. This information is presented and summarized in Table 3.

As can be seen from the table, nearly all of the specifications were completely met, with some of the less important specifications being partially met.

Assessment of Significance

The device has a significant impact on the ease of use and clarity of data for this type of experimental procedure. It is a tool that allows for a more rigorous data analysis to give a better understanding of the opioid self-administration patterns in mice. It confirms what had been previously determined, that there is significant feeding during the mouse's active cycle, and it elucidates that the presence of morphine, CBD, and THC has no effect on feeding frequency. However, it demonstrated that there was a significant difference between the amount of gel consumed for THC as compared with morphine or CBD. Therefore, it shows that while the presence of THC did not affect the feeding frequency of the mice, it did affect the amount they were consuming during each bout. This could have important applications in understanding the effect of analgesics among patients.

In addition, the pain tests performed demonstrated that the mice in the morphine group eventually built up a tolerance to the drug, while the mice in both the THC and CBD group did not. This could have important implications, as they could act as analgesics without the negative effect of building up tolerance, which is associated with morphine.

Assessment of Impact

This device has a significant impact in terms of experimental procedure. It is an easy to use, accurate, and robust design. In addition, it provides more accurate information about the feeding profile of the different mice. Previous to this device, only the net amount of gel consumed could be determined. However, as a result of this device, it allows for continuous monitoring of gel left in the cage and can determine exactly when and how much the mouse is feeding. This provides a more complete understanding of how different drugs or other substances affect the mouse self-administration.

In addition to investigating the effect of morphine, the device was used to assess the impact of CBD and THC as analgesics for treatment of chronic pain[10]. Since both NSAIDs and opioids have significant side effects and limitations, it has caused increased scrutiny as to alternative painkillers, such as CBD and THC. This device helped to show how these drugs affect consumption patterns, and thus any possibility of developing a substance use disorder, and how effective their analgesic properties are.

The device can be adapted to be used for a variety of other experimental designs. For example, it could be used in experimental designs involving alcohol or ethanol self-administration.

Future Work

While this project is a great aid in the experimental procedure, there are still more future steps that could be taken to improve upon the design. For example, RFID technology could be used to track individual mice and when they approach the feeder for self-

administration. In this instance, multiple mice could be housed in a single cage and could be tracked through an implantable RFID chip.

In addition, information could be collected about the body temperature, respiratory rate, and other biometrics of the mice analyze how opioids can affect behavior and other features.

Ultimately, it is important to not lose sight of the end goal, which is to help individuals suffering from opioid addiction. This device is a tool that allows for a better experimental design to be carried out and understand how opioids could affect human behavior, through a rodent model. It will increase the capability of future experimenters to rigorously perform these types of experiments. This will ultimately help elucidate information about key components in the signaling cascades that result in addiction and will help to develop new methods for blocking those types of pathways while still retaining the analgesic effects of these drugs.

Acknowledgements

Special thanks to my PI, Charles Chavkin, PhD, my lab mentors, Antony Abraham, PhD, and Benjamin Land, PhD, and my Bioengineering advisor, Chris Neils, PhD.

References

1. CDC/NCHS, National Vital Statistics System, Mortality. CDC Wonder, Atlanta, GA: US Department of Health and Human Services, CDC; 2017. https://wonder.cdc.gov.
2. Florence CS, Zhou C, Luo F, Xu L. The Economic Burden of Prescription Opioid Overdose, Abuse, and Dependence in the United States, 2013. *Med Care*. 2016;54(10):901-906. doi:10.1097/MLR.0000000000000625.
3. Wikler, Abraham. "Conditioning factors in opiate addiction and relapse." *Journal of Substance Abuse Treatment* 1.4 (1984): 279-285.
4. Manglik, Aashish, et al. "Structure-based discovery of opioid analgesics with reduced side effects." *Nature* 537.7619 (2016): 185.
5. Hayar, Abdallah, et al. "A low-cost solution to measure mouse licking in an electrophysiological setup with a standard analog-to-digital converter." *Journal of neuroscience methods* 153.2 (2006): 203-207.
6. Wikler, Abraham, and Frank T. Pescor. "Classical conditioning of a morphine abstinence phenomenon, reinforcement of opioid-drinking behavior and “relapse” in morphine-addicted rats." *Psychopharmacologia* 10.3 (1967): 255-284.
7. Weeks, James R. "Experimental morphine addiction: method for automatic intravenous injections in unrestrained rats." *Science* 138.3537 (1962): 143-144.
8. Hill, William A. "Public Health Service policy on humane care and use of laboratory animals." (1986).
9. ASTM International. *D5927-17 Standard Classification System for and Basis for Specifications for Thermoplastic Polyester (TPES) Injection and Extrusion Materials Based on ISO Test Methods*. West Conshohocken, PA, 2017. Web. 3 Dec 2018. <https://doi.org/10.1520/D5927-17>
10. Leung, Edward JY, et al. "Orally consumed cannabinoids provide long-lasting relief of allodynia in a mouse model of chronic neuropathic pain." *bioRxiv* (2019): 556373.
11. Seidle, Nathan (2014) SparkFun_HX711_Calibration (Version 2.0) [Source Code]. https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide

Appendix A

Load Cell Calibration Sketch

```
#include "HX711.h"
#include <LiquidCrystal.h>

#define DOUT  6
#define CLK  7
LiquidCrystal lcd(9, 8, 5, 4, 3, 2);

HX711 scale(DOUT, CLK);

float calibration_factor = 7530;                                    //=100g
float output;
int readIndex;
float total=0;
float average=0;
float average_last=0;
const int cycles=20;
float readings[cycles];

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.set_scale();
  scale.tare();  //Reset the scale to 0

  long zero_factor = scale.read_average();                          //Get a baseline reading
  Serial.print("Zero factor: ");                                    //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
  lcd.begin(16, 2);
}

void loop() {

  scale.set_scale(calibration_factor);                              //Adjust to this calibration factor

  Serial.print("Reading: ");
  output=scale.get_units(), 2;
 lcd.print(calibration_factor);
 lcd.setCursor(0, 1);
 lcd.print(output);


//Smoothing the results
 // subtract the last reading:
 total = total - readings[readIndex];
 // read from the scale
 readings[readIndex] = scale.get_units(), 2;
 // add the reading to the total:
 total = total + readings[readIndex];
 // advance to the next position in the array:
 readIndex = readIndex + 1;

 // if we're at the end of the array...
 if (readIndex >= cycles) {
   // ...wrap around to the beginning:
   readIndex = 0;
 }
 // calculate the average:
 average = total / cycles;

 average=scale.get_units(), 2;


//Zero drift compensation
 if((average_last>average+0.03 || average_last<average-0.03)){
     //Minimum Load defines Zero-Band
     if (average<0.06){average=0;}
     Serial.print("\tFilter: ");
     Serial.print(average);
     average_last=average;
 }
 else{
   Serial.print("\tFilter: ");
   Serial.print(average_last);
 }
 Serial.print(" g");                                                //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
 Serial.print(" calibration_factor: ");
 Serial.print(calibration_factor);
 Serial.println();

 if(Serial.available())
 {
   char temp = Serial.read();
   if(temp == '+' || temp == 'a')
     calibration_factor += 10;
   else if(temp == '-' || temp == 'z')
     calibration_factor -= 10;
 }
}
```

Load Cell Measurement Sketch

```
// Brenden Wong
// Chavkin Lab
// SURP 2018

//This sketch allows Arduino Uno to record the mass of a gel cup and write the resulting mass along with a time stamp to an SD card

# include <Time.h>                              // include time library
# include <TimeLib.h>
# include "HX711.h"                             // include load cell library
# include <SD.h>                                // include capability to write to SD card
# include <SPI.h>
# include <LiquidCrystal.h>
# define DOUT 6                                 // define pins for load cell
# define CLK 7

LiquidCrystal lcd(9, 8, 5, 4, 3, 2);

HX711 scale(DOUT, CLK);                         //////////////////////
float calibration_factor = 7530; //<----------///Adjust as needed///
char filename[] = "LOADCELL.CSV";               //////////////////////
char filenameTime[] = "TIME.CSV";
int chipSelect = 10;
File file;
File fileTime;

void setup() {
  Serial.begin(9600);
  pinMode(chipSelect, OUTPUT);                      // chip select pin must be set to OUTPUT mode
  if (!SD.begin(chipSelect)) {                      // Initialize SD card
    Serial.println("Could not initialize SD card."); // if return value is false, something went wrong.
  }
  if (SD.exists(filename)) {                        // if file exists, file will be deleted
    Serial.println("File exists.");
    if (SD.remove(filename) == true) {
      Serial.println("Successfully removed file.");
    } else {
      Serial.println("Could not removed file.");
    }
  }
  if (SD.exists(filenameTime)) {
    Serial.println("Time exists.");
    if (SD.remove(filenameTime) == true) {
      Serial.println("Successfully removed time file.");
    } else {
      Serial.println("Could not remove time.");
    }
  }
  scale.set_scale();
  scale.tare();                                     //Reset the scale to 0
  long zero_factor = scale.read_average();          //Get a baseline reading
  lcd.begin(16,2);
  lcd.print("Mass is:");
}

void loop(){
  scale.set_scale(calibration_factor);              //Adjust to this calibration factor
  file = SD.open(filename, FILE_WRITE);
  if (file) {
    file.println(scale.get_units(), 2);             // write number to file
    file.close(); // close file
    Serial.print("Wrote number: ");                 // debug output: show written number in serial monitor
    Serial.println(scale.get_units(), 2);
  } else {
    Serial.println("Could not open file (writing).");
  }
  fileTime = SD.open(filenameTime, FILE_WRITE);
  if (fileTime) {
    fileTime.print(hour());
    fileTime.print(":");
    fileTime.print(minute());
    fileTime.print(":");
    fileTime.println(second());
    fileTime.close();
    Serial.print("Wrote time: ");
    Serial.print(hour());
    Serial.print(":");
    Serial.print(minute());
    Serial.print(":");
    Serial.println(second());
  } else {
    Serial.println("Could not write time");
  }
  Serial.print("Scale reading: ");
  Serial.println(scale.get_units(), 2);
  lcd.setCursor(0, 1);
  lcd.print(scale.get_units(), 2);
  delay(2000);                                      // Delay 2 seconds
}
```

Appendix B

Load Cell Processing Script

```
clear all; close all; clc
%% Read in and process initial mass and time data
prompt = 'Enter Date (YYMMDD): '; % prompt user for time data
mousePrompt = 'Enter Mouse Number (##): ';
file = input(prompt, 's');
mouseNumber = input(mousePrompt, 's');
weight = csvread(strcat(file, '_', mouseNumber, 'LOADCELL.CSV'));
fid = fopen(strcat(file, '_', mouseNumber, 'TIME.CSV'));
A=textscan(fid,'%8s');
dn=datenum(A{1},'HH:MM:SS');
if (size(weight, 1) ~= size(dn, 1))                    % if the dimensions are not the same, modify the time data
    dn = dn(1:end - 1);
end

index = 1;
difference = 10;
while weight(index) < 0.02 || abs(difference) > 0.02   % cut off initial data
    difference = weight(index + 1) - weight(index);
    index = index + 1;
end
dn = dn(index : end);
weight = weight(index : end);

n = zeros(size(weight));
for i = 1 : size(n)
    n(i) = i / 30 / 60;
end

%% Plot raw consumption data
figure(1)
plot(n, weight, 'o');
title('Gel Consumption', 'FontSize', 18);
ylabel('Mass (g)', 'FontSize', 16);
xlabel('Time', 'FontSize', 16);
legend('Mass of Gel in Cup', 'Location', 'SouthEast');

%% max out distinct eating events
newWeight = weight;                                    % new vector for thresholded data
cap = 20;                                              % arbitrary maximum value to set thresholded data to
for i = 1 : size(weight) - 1                           % parse through weight vector
    diff = weight(i + 1) - weight(i);                  % compare data at each index with the next one
    if abs(diff) > 0.04 || weight(i) > cap             % if data points are unstable, max out data
        newWeight(i + 1) = cap;
    else
        newWeight(i) = weight(i);                      % if data is stable, keep original weight data
    end
end

%% Plot new processed data
figure(2)
fig = plot(n, newWeight,'ko');
title('Gel Consumption Adjusted (Relative Time)', 'FontSize', 18)
ylim([-5 cap + 5])
ylabel('Mass (g)', 'FontSize', 16)
xlabel('Time (hours)', 'FontSize', 16)
legend(strcat('Mouse Number: ', mouseNumber), 'Location', 'SouthEast')
saveas(fig, strcat(file, '_', mouseNumber, 'ProcessedData.jpg'))
figure(2)
plot(n, newWeight,'ko')
title('Gel Consumption Adjusted (Relative Time)', 'FontSize', 18)
ylim([-5 cap + 5])
ylabel('Mass (g)', 'FontSize', 16)
xlabel('Time (hours)', 'FontSize', 16)
legend('Mass of Gel in Cup', 'Location', 'SouthEast')

```

```
%% Separating Data
n = n(300 : end);
newWeight = newWeight(300 : end);                          % Clip new data
inc = 1;
while newWeight(1) == 20                                   % Eliminate any non-mass readings
   newWeight = newWeight(2 : end);
   n = n(2 : end);
end
eatingEvents = zeros(size(newWeight));
massData = zeros(size(newWeight));
for i = 1 : size(newWeight)                                % Loop through and classify each point as an eating event or
                                                           % a mass value
    if newWeight(i) > 19.9  || newWeight(i) < 0
        if i > 1
            eatingEvents(i) = 1;
            massData(i) = massData(i - 1);
        end
    else
        massData(i) = newWeight(i);
    end
end
massData = massData(1) - massData;                         % Change the data to reflect mass consumed instead of mass remaining
for i = 1 : size(massData)
   if massData(i) < 0 && i ~= 1
      massData(i) = massData(i - 1);
   end
end
testMassData = massData;
for i = 1 : size(testMassData)                             % Account for gel dessication rate
    testMassData(i) = testMassData(i) - ((1.06 * 10 ^ -5) * 2 * i);
end
windowSize = 500;                                          % Smooth the resulting mass curves
b = (1/windowSize)*ones(1,windowSize);
a = 1;
massData = filter(b, a, massData);

figure(4)
subplot(2,1,1)
plot(n, massData, 'ko', 'MarkerSize', 3);
title('Mass of Gel Consumed', 'FontSize', 18)
ylabel('Mass of gel (g)')
xlabel('Time (hours)')

subplot(2,1,2)
plot(n, eatingEvents, 'ko');
ylim([0.99 1.1]);
title('Eating Events', 'FontSize', 18)
xlabel('Time (hours)');

%% Create distribution of inter-eating intervals
binsize = 5;            %<----------------------------- size of bins

threshold = 120;        %<-- time in seconds between events required to consider distinct eating event

bouttimes = [];
endEvent= 0;
bouts = [];
for i = 1 : size(eatingEvents)                    % Count the number of bouts and stamp when they occur
    if (eatingEvents(i) == 1)
        diff = n(i) - endEvent;
        if (diff * 3600 >= threshold)
           bouts = [bouts; endEvent diff * 60];
        end
        endEvent = n(i);
        bouttimes = [bouttimes; diff];
    end
end
bouttimes = bouttimes * 3600;
bins = 0 : binsize : max(bouttimes);
bins =[zeros(size(bins)); bins];
bouts
xlswrite(strcat(file, '_', mouseNumber, 'boutData.xlsx'), bouts(:,1))
numberOfBouts = size(bouts,1)

% Plot frequency distribution of inter-eating intervals
figure(5)
histogram(bouttimes, size(bins,2))
ylim([0 50])
xlim([0 250])
xlabel('Inter-eating interval (sec)', 'FontSize', 14)
ylabel('Number of intervals', 'FontSize', 14)
title('Frequency Distribution of Inter-eating Intervals','FontSize', 18)
ylabel('Average Gel Consumption in 5 Minute Period (g/sec)', 'FontSize', 18)
hold on
plot(t, vals, 'r-', 'LineWidth', 2)
legend('Correlation', 'Linear Regression Line')
text(50, 2*10^-4, ['R^2 = ' num2str(model.Rsquared.Adjusted)], 'FontSize', 16)
text(50, 0*10^-4, ['p = ' num2str(model.Coefficients{2,4})], 'FontSize', 16)

%% Save processed Data to a File
endPrompt = 'Would you like to save the data? (y or n) : ';
if (input(endPrompt, 's') == 'y')
   xlswrite(strcat(file, '_', mouseNumber, 'loadcellData.xlsx'), massData);
end
```

Appendix C

Scale Assembly Instructions

The following sections describe methods for manufacturing and using the scale.

Stage 1: Acquiring Required Parts for Scale

The following is a list of all the components you will need to construct the scale:

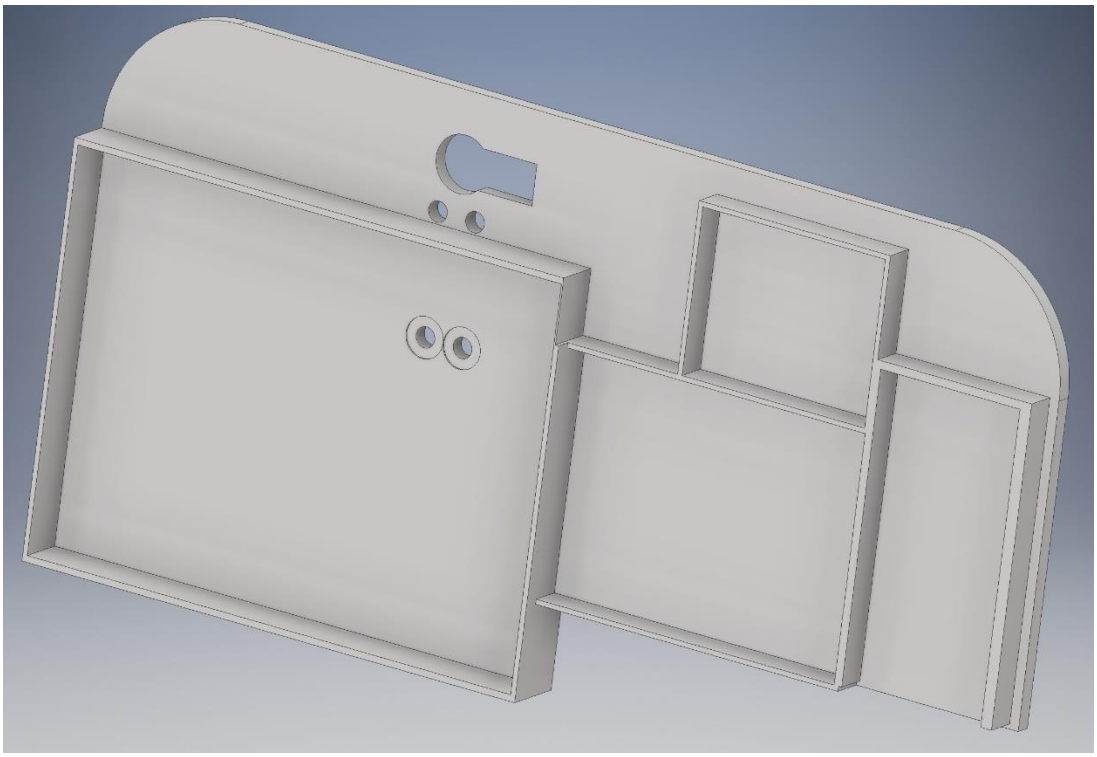
- 1 Arduino Uno
- 1 HX711 Load Cell Amplifier
- 1 100 g Load Cell
- Jumper wires
- 1 Mini breadboard
- 1 microSD Card Reader Module
- 1 MicroSD Card
- USB Printer Cable
- 8 M3 12 mm screws
- Gel cup

You will need the following tools:

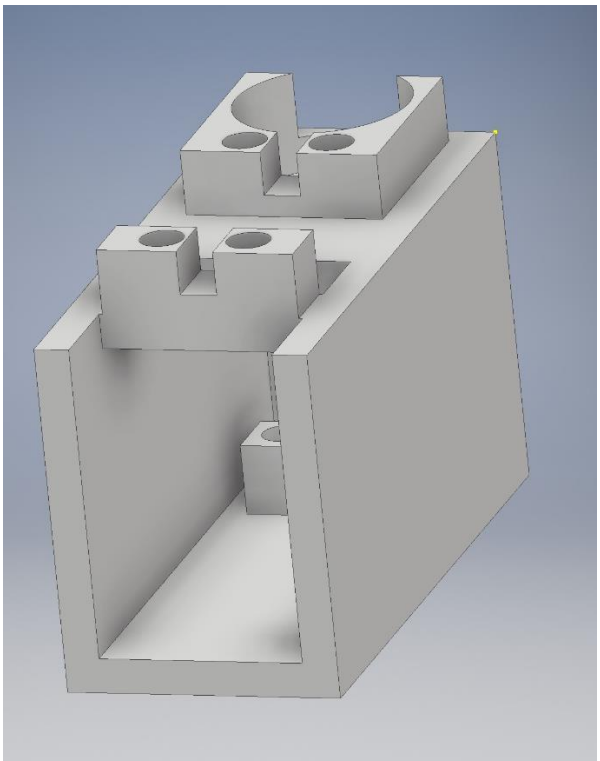
- Soldering iron
- Super glue
- Heat gun
- Heat shrink tubing

You will also need to 3D print the following pieces:

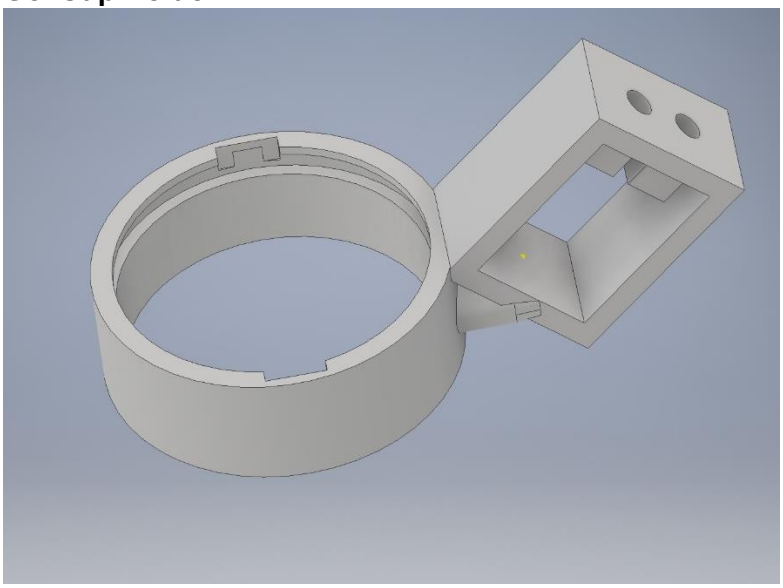
- **Base Plate**



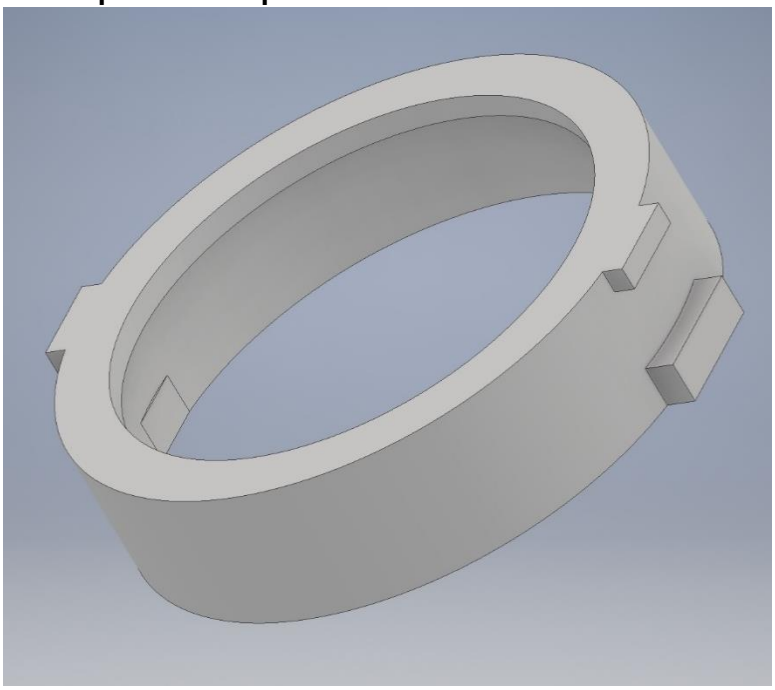
- **Base Plate Bottom**



- **Gel Cup Holder**



- **Gel Cup Holder Top**



The CAD files for the 3D prints can be found in the supplementary materials. Instructions on how to assemble the circuit will follow later.

A table listing where you can find certain components can also be found in the supplementary materials.

Stage 2: Preparing the Components

The load cell and the load cell amplifier will both require some soldering and other modifications.

The load cell amplifier will need to have the breakaway header pins soldered on and 90 degree breakaway headers may be used for the load cell pins to facilitate wired connections in the circuit, but are not necessary.

The leads on the load cell are very fragile and must be handled carefully.

1. Use superglue on the base of the load cell and the wires to strengthen that area, since it is the most fragile and likely to break.
2. Strip the ends of the leads on the load cell so that about half an inch of wire is exposed.
3. Take two female-female jumper wires and cut them both in half and strip the ends of the wires so that about half an inch of wire is exposed.
4. Wrap the exposed wire from each of the load cell leads with the exposed wire of one of the halves of the jumper wires.
5. Use a soldering iron to solder each of the connections together.
6. Use the small heat shrink tubing to cover all the exposed wire for the individual load cell leads.
7. Use the large heat shrink tubing to group all four of the soldered connections together, leaving about a quarter of an inch of the bottom of the small heat shrink tubing exposed for each of the leads.

To prepare the gel cup itself, fit the gel cup inside of the gel cup holder top piece. It should fit tightly but may require some additional superglue to secure the connection. It should be able to fit inside the gel cup holder piece. Once you put the gel cup in the holder, rotate it 90 degrees to lock it in place. It should still be able to rotate but will not removable from the gel cup. To remove it, simply twist until the tabs of the gel cup holder top piece are aligned with the notches in the gel cup holder.

Stage 3: Scale Assembly

1. Place the baseplate on the wire cage top
2. Insert the wired end of the load cell in the base plate bottom and align the holes of the load cell with the holes on the bottom of the base plate bottom piece. Insert screws through the base plate bottom holes and the load cell holes, ensuring that the load cell remains level.
3. Feed the wires from the load cell through the large hole in the top of the base plate bottom
4. Place the other end of the load cell underneath the holes from the gel cup holder and align the gel cup holder holes with the load cell holes
5. Insert screws from the top of the gel cup holder.
6. From underneath the wire cage top, align the four holes and the large hole with the wires from the base plate bottom piece with the four holes and the large hole

in the base plate. Feed the wires from the load cell through the large hole of the base plate.

7. Insert screws into the four holes in the base plate, ensuring that the base plate bottom piece remains level.

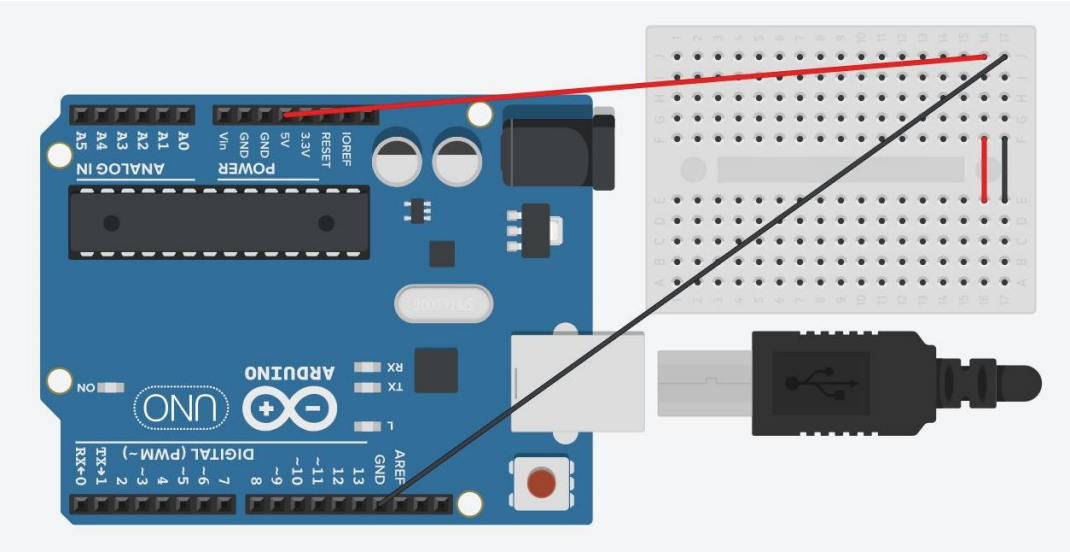
The scale assembly should be secure on the wire rack.

Stage 4: Configuring the Circuit

Make the following connections on the circuit components:

Arduino and Breadboard:

Since multiple components will require a positive power supply and ground pins, make positive and ground power rails on the breadboard. The following figure is one suggestion of how to do this.



Load Cell Amplifier:

Connect:

Load Cell Amplifier -----> Arduino
VCC ----------> Vcc
DAT----------> Pin 6
CLK----------> Pin 7
GND ----------> Gnd

Load Cell Amplifier ----> Load Cell
RED ----------> Red wire
BLK ----------> Black wire

WHT ----------> White wire
GRN ----------> Green wire

Use jumper wires and the breadboard as necessary to make the connections with the Arduino. Note that to reduce the number of wires, the load cell amplifier headers could be connected directly to the breadboard and use the breadboard pins to connect to the Arduino.

SD Card Reader Module:

Connect:

SD Card Module ------> Arduino
CS ----------> Pin 10
SCK ----------> Pin 13
MOSI ----------> Pin 11
MISO ----------> Pin 12
VCC ----------> Vcc
GND ----------> Gnd

Use jumper wires and the breadboard as necessary to make the connections with the Arduino.

Liquid Crystal LED Display:

Connect:
LED Display ----------> Arduino
GND ----------> Gnd
VCC ----------> Vcc
V0 ----------> Gnd
RS ----------> Pin 9
R/W ----------> Gnd
E ----------> Pin 8
DB4 ----------> Pin 5
DB5 ----------> Pin 4
DB6 ----------> Pin 3
DB7 ----------> Pin 2
B+ ----------> Vcc
B- ----------> Gnd

Use jumper wires and the breadboard as necessary to make the connections with the Arduino.

Stage 5: Calibrating the Scale

Upload the load cell calibration sketch to the Arduino, ensuring that the sketch is using the correct serial port for the Arduino. For the full calibration sketch, see Appendix A and

can also be found in the supplementary materials. Open the serial monitor on the Arduino IDE and put a known mass in the gel cup. Adjust the calibration factor until the displayed value on the serial monitor matches the actual mass. The calibration factor should be around 6720 but may vary slightly for individual load cells.

Stage 6: Load Cell Sketch

Modify the calibration factor in the load cell sketch to match the determined calibration factor from the previous stage. For the full load cell sketch, see Appendix B and can also be found in the supplementary materials. Simply upload the code to the Arduino, ensuring that the sketch is using the correct serial port for the Arduino.

Stage 7: Operating the Scale

1. Remove the microSD card from the module and move data files "LOADCELL.CSV" and "TIME.CSV" to a computer
    a. Rename "LOADCELL.CSV" and "TIME.CSV" by inserting the date (YYMMDD) and the mouse number (##). For example, removing the card from mouse one's cage on August 7, 2018, we would rename the load cell file as "180807_01LOADCELL.CSV" and the time file as "180807_01TIME.CSV".
2. Insert the microSD card back in the module
3. Remove the gel cup from the holder
4. Record the mass of gel and load it with new gel and record the new weight
5. Ensure that the USB printer cable is connected to a power source.
6. Press the reset button on the Arduino to initialize the data collection and to tare the scale.
7. Secure the gel cup in the holder
8. Replace the wire and plastic cage tops, being careful not to shift the wiring too much

Stage 8: Processing the Data

For the full MATLAB script, see Appendix C and can also be found in the supplementary materials. Ensure that the MATLAB script and the loadcell and time files are in the same directory. Simply run the MATLAB script and follow the prompts on the command window. Note that the files for each of the load cell data and time data must be named exactly in the style of YYMMDD_##LOADCELL.CSV and YYMMDD_##TIME.CSV without any spaces for the program to work. If you get an error, check to see if the files are named properly. When the script finishes running, enter "y" on the command line to save the processed mass data.

After saving the mass data to the same directory as the MATLAB script, move the mass data to the folder containing data for the specific mouse. Then run the corresponding MATLAB script and follow the prompt to enter how many days of data are being processed to average the mass data over several days. The MATLAB script will automatically save the averaged mass data into the same directory as the script.

After saving the average data for each of the mice, put the saved average data for each of the mice into the same directory as the GelAverages MATLAB script and run it. This will produce figures comparing each of the mice in each group of the drugs individually and combined with the other drugs

7.2 Appendix II: Code Documentation

7.2.1 *Ex vivo* Single Cell Imaging Example Code

```
clear all; close all; clc
folderPath = 'E:\Lab\Cell Imaging\Fixed Cell Imaging\091019 kr kn hyperred
slice 2 NALFURAFINE';
cd(folderPath)

% Read in files in data folder
files = dir('*.mat');
data = cell(length(files), 1);
for i = 1 : length(files)
    temp = open(files(i).name);
    data{i} = temp.images;
end
filename = files(1).name(1 : end - 4);

numFrames = length(data{1});
for i = 1 : numFrames
   imshow(data{1}{i})
   pause(0.1)
   disp(strcat('Reading frame', {' '}, num2str(i)))
end
TREATMENTS = [7];

% Define baseline frames
baselineFrames = 4 : 7;

% Initialize signal structure
signal = cell(length(numFrames), 1);

% Initizalize cell for holding binarized locations
filled = cell(length(numFrames), 1);

% Initialize cell for holding cell borders
borders = cell(length(numFrames), 1);

% Loop through baseline frames
for k = baselineFrames
    frame = data{1}{k};
    filled{k} = false(512);

    % Find circles within radius range and sensitivity level
    [centers,radii] = imfindcircles(frame,[4,15],'Sensitivity',.90);
    signal{k} = zeros(length(centers), 1);

    % Loop through each cell
    for i = 1 : length(centers)
        radius = floor(radii(i));
```

```
        % Create structuring element with specified radius
        SE = strel('disk', radius, 0);

        % Find indices that correspond with the structuring element
        circle = find(getnhood(SE));

        % Create X and Y coordinates about center for circle of interest
        [X,Y] = ind2sub(size(getnhood(SE)), circle);
        X = X - radius - 1;
        Y = Y - radius - 1;

        % Ensure that there is no overlap with the edge of the frame
        if floor(centers(i, 2)) + min(X) > 0 && ceil(centers(i, 2)) + max(X) <
512
            if floor(centers(i, 1)) + min(Y) > 0 && ceil(centers(i, 1)) +
max(Y) < 512
                cellIndex = sub2ind(size(frame), ceil(centers(i,2)) + X,
ceil(centers(i,1)) + Y);
                filled{k}(cellIndex) = true;
                borders{k} = bwperim(filled{k});
                signal{k}(i,1) = sum(frame(cellIndex));
            end
        end
    end
end
compositeBaselineFrame = false(512);

for k = baselineFrames
   compositeBaselineFrame(filled{k}) = true;
end

% Create structure for the location for each cell
cc = bwconncomp(compositeBaselineFrame, 4);
baselineCellIdx = cc.PixelIdxList';

stat = regionprops(cc, 'centroid', 'area');
baselineCellLocs = zeros(length(stat), 2);
for i = 1 : length(stat)
   baselineCellLocs(i, :) = stat(i).Centroid;
end

% Generate baseline fluorescence value for each region of interest
baselineCellFluor = zeros(length(baselineCellIdx), length(baselineFrames));
for k = baselineFrames
    frame = data{1}{k};
    for i = 1 : length(baselineCellIdx)
       baselineCellFluor(i, k - baselineFrames(1) + 1) = sum(frame(baselineCel-
lIdx{i})) / stat(i).Area;
    end
end
baselineCellFluor = mean(baselineCellFluor, 2);

% Generate fluorescence values for each region of interest for each frame
cellFluor = zeros(length(baselineCellIdx), numFrames);
for k = 1 : numFrames
   frame = data{1}{k};
   for i = 1 : length(baselineCellIdx)
      cellFluor(i, k) = sum(frame(baselineCellIdx{i})) / stat(i).Area;
   end
   cellFluor(:, k) = (cellFluor(:, k) - baselineCellFluor) ./ baselineCellFluor
* 100;
```

```
end
cellFluor = cellFluor';
minAbsDist = zeros(size(cellFluor, 2), 1);

for i = 1 : size(cellFluor, 2)
   if i ~= size(cellFluor, 2)
      minAbsDist(i) = sum(abs(cellFluor(:, i) - cellFluor(:, i + 1)).^2);
   else
      minAbsDist(i) = sum(abs(cellFluor(:, i) - cellFluor(:, 1)).^2);
   end
   for j = 1 : size(cellFluor, 2)
      if j ~= i
          tempDist = sum(abs(cellFluor(:, i) - cellFluor(:, j)).^2);
          minAbsDist(i) = min(minAbsDist(i), tempDist);
      end
   end
end
fig = figure;

dist = minAbsDist > 160;

hold on
for i = 1 : length(baselineCellIdx)
   txt = char(strcat('Cell', {' '}, num2str(i)));
   if dist(i) == 1
      plot(cellFluor(:, i), '-o', 'DisplayName', txt, 'LineWidth', 2)
   end
end

for i = 1 : length(baselineCellIdx)
   if dist(i) == 0
      plot(cellFluor(:, i), '-o', 'LineWidth', 0.5)
   end
end

for i = 1 : length(TREATMENTS)
   txt = char(strcat('Treatment', {' '}, num2str(i)));
   line([TREATMENTS(i) TREATMENTS(i)], [-60 40], 'DisplayName', txt, 'Lin-
eWidth', 2, 'Color', 'k')
end

% legend show
% lgd = legend('Location', 'BestOutside');
% lgd.FontSize = 8;
box on
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
filename = 'Individual Nalfurafine Traces';
title(filename, 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

%saveas(fig, strcat(filename, '.jpg'))
exportFluor = [(1 : size(cellFluor, 1))' cellFluor];
% xlswrite(strcat(filename, '.xlsx'), exportFluor, 1, 'A2')
% xlswrite(strcat(filename, '.xlsx'), {'Frame Number'}, 1, 'A1')

figure;
```

```
imshow(imoverlay(data{1}{1}, bwperim(compositeBaselineFrame)))

hold on
for i = 1 : length(baselineCellIdx)
   cellDescrip = strcat('Cell', {' '}, num2str(i), ':', {' '},
num2str(round(baselineCellFluor(i))));
   text(baselineCellLocs(i, 1) + 5, baselineCellLocs(i, 2) + 5, num2str(i) ,
'color', 'red', 'FontSize', 8)
end

baselineFrameFluor = zeros(length(baselineFrames), 1);

for k = baselineFrames
   baselineFrameFluor(k - baselineFrames(1) + 1) = sum(sum(data{1}{k}));
end

frameFluor = zeros(length(baselineFrames), 1);
for k = 1 : numFrames
   frameFluor(k) = (sum(sum(data{1}{k})) - mean(baselineFrameFluor)) /
mean(baselineFrameFluor) * 100;
end
figure;

plot(frameFluor, '-o', 'LineWidth', 2)
hold on
for i = 1 : length(TREATMENTS)
    line([TREATMENTS(i) TREATMENTS(i)], [-40 40], 'color', [i * 30 + 50 100 * i
- 70 50] / 255, 'LineWidth', 1.5)
end
legend('Frame Fluorescence', 'Treatment 1')
```

7.2.2 Single HEK Cell Imaging Load Gifs Code

```
clear all; close all; clc
folderPath = 'E:\Lab\Cell Imaging\Live Cell Imaging\Data\191115';
cd(folderPath);

files = dir('**');
files(1 : 2) = [];

numFiles = numel(files);

data = cell(numFiles, 1);
for i = 1 : numFiles
    filename = files(i).name;
    data{i} = imread(filename, 'gif');
    disp(strcat('Reading File: ', {' '}, filename))
end
numFrames = zeros(numFiles, 1);

for i = 1 : numFiles
   numFrames(i) = size(data{i}, 4) / 2;
end
```

```
img_hyper = cell(numFiles, 1);

for i = 1 : numFiles
   img_hyper{i}(:, :, :) = data{i}(:, :, 1, 2 : 2 : size(data{i}, 4));
end

img_hy31 = cell(numFiles, 1);
for i = 1 : numFiles
   img_hy31{i}(:, :, :) = data{i}(:, :, 1, 1 : 2 : size(data{i}, 4));
end

filename = files(1).name;
filename = filename(1 : 6);
%save(strcat(filename, '.mat'), 'img_hyper', 'img_hy31', 'numFrames')
```

7.2.3 Single HEK Cell Imaging Example Code

```
clear all; close all; clc
plateNumber = 2;

positions = 1 : 4;



individual_cell_data = cell(length(positions), 1);

originalFileName = '190821';
maskFileName = strcat('BPF_', originalFileName);
data_original = open(strcat(originalFileName, '.mat'));
data = open(strcat(maskFileName, '.mat'));

img_hrm63_raw = data_original.img_hyper;
img_hrm63_mask = data.img_hyper;

% img_hrm63_raw = data_original.img_hrm63;
% img_hrm63_mask = data.img_hrm63;

numFrames = data.numFrames;

numFiles = length(img_hrm63_raw);
pos = 1;

positionNumber = positions(pos);
dataNumber = (plateNumber - 1) * 4 + positionNumber;
img_hrm63 = img_hrm63_mask;



baselineFrames = 5 : 10;
```

```
numCells =      zeros(numFrames(dataNumber), 1);

pixList =       cell(numFrames(dataNumber), 1);

filledFrames =  cell(numFrames(dataNumber), 1);

centroidLocs =  cell(numFrames(dataNumber), 1);

cellAreas =     cell(numFrames(dataNumber), 1);

newBorders =    cell(numFrames(dataNumber), 1);

newPixList =    cell(numFrames(dataNumber), 1);




for k = baselineFrames
    frame1 = img_hrm63{dataNumber}(:, :, k);
    sub_mask = frame1;
    threshold = mean(mean(sub_mask));
    sub_mask(sub_mask >=  threshold) = threshold;
    sub_mask = imgaussfilt(sub_mask);

    frame1 = frame1 - sub_mask;
    frame1 = imgaussfilt(frame1);
    borders = zeros(512,512);

    dist1 = 1;
    threshUpper = 15;
    threshLower = 15;
    for r = 1 + dist1 : 512 - dist1
        for c = 1 + dist1 : 512 - dist1
            p    = frame1(r, c);
            pU1  = frame1(r - dist1, c);
            pL1  = frame1(r, c - dist1);
            pR1  = frame1(r, c + dist1);
            pB1  = frame1(r + dist1, c);

            AOI = [p pU1 pB1 pL1 pR1];

            if sum(AOI) ~= 0
                if p >= threshUpper
                    if ((pU1 <= threshLower && pB1 >= threshUpper) || (pU1 >=
threshUpper && pB1 <= threshLower))
                        borders(r,c) = 1;
                    elseif (pL1 <= threshLower && pR1 >= threshUpper) || (pL1
>= threshUpper && pR1 <= threshLower)
                        borders(r, c) = 1;
                    end
                end
            end
        end
    end

    bordersMem = bwmorph(borders, 'close');
```

```
    se = strel('disk', 2);
    borders = imclose(bordersMem, se);

    for i = 1 : 512
       for j = 1 : 512
          if i == 512 || i == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif j == 512 || j == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif i == 512 && j == 512
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif i == 1 && j == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          end
       end
    end
    filled = imfill(borders,'holes');

    [labeledImage, numberOfRegions] = bwlabel(filled);
    labeledImage = bwareaopen(labeledImage, 100);

    binaryLabIm = zeros(512);
    binaryLabIm(labeledImage) = 1;
    borders(~labeledImage) = 0;


    filled = imfill(borders, 'holes');
    D = -bwdist(~filled);
    mask = imextendedmin(D, 2);
    D2 = imimposemin(D, mask);
    Ld = watershed(D2);
    filled(Ld == 0) = 0;
    L = filled;

    imshow(L)
    filledFrames{k} = L;

    % Define structure of connected components
    cc = bwconncomp(L, 4);
    numCells(k) = cc.NumObjects;
    pixList{k} = cc.PixelIdxList;

    % Extract centroid and area for each ROI
    stat = regionprops(cc, 'centroid', 'area');
    newPixList{k} = cell(size(pixList{k}));
    for i = 1 : length(stat)
        % Check if cell is within frame
        if stat(i).Centroid(1) < 510 && stat(i).Centroid(2) < 510 &&
stat(i).Centroid(1) > 10 && stat(i).Centroid(2) > 10

           % Size exclusion for cells smaller than 10 pixels
           if stat(i).Area > 10
              % Store centroid locations, cell areas, and pixel index list
```

```
                centroidLocs{k} = [centroidLocs{k}; stat(i).Centroid];
                cellAreas{k} = [cellAreas{k}; stat(i).Area];
                newPixList{k}{1, i} = pixList{k}{1, i};
            end
        end
    end
    newPixList{k} = newPixList{k}(~cellfun('isempty', newPixList{k}));
end

for i = baselineFrames
    newBorders{i} = bwperim(filledFrames{i});
end

compositeBaselineFrame = false(size(filled));
for k = baselineFrames
   for j = 1 : length(newPixList{k})
      compositeBaselineFrame(newPixList{k}{j}) = true;
   end
end

adj_pixList = cell(numFrames(dataNumber), 1);

% Remove empty indices from the pixel index list
for k = baselineFrames
    adj_pixList{k} = newPixList{k}(~cellfun('isempty', newPixList{k}));
end
baselineCentroids = centroidLocs{baselineFrames(1)};

baselinePix = adj_pixList{baselineFrames(1)};

duplicates = cell(length(baselineCentroids), 2);

count = 1;



% Loop through length of baseline centroids
for i = 1 : length(baselineCentroids)
   % Check if each baseline centroid is within 0.03 of any other centroid
   if any(ismembertol(baselineCentroids(i, :), baselineCentroids, 0.03, 'By-
Rows', true))
       % Return index/indices where this occurs
       [~, index] = ismembertol(baselineCentroids(i, :), baselineCentroids,
0.02, 'ByRows', true', 'OutputAllIndices', true);

       % Store original baseline centroid index
       duplicates{count, 1} = i;
       % Store index/indices within tolerance of original baseline centroid
       duplicates{count, 2} = index{1};
       count = count + 1;
   end
end


usedPairs = [0 0];
% Loop through length of baseline centroids
for i = 1 : length(duplicates)
   % If there are more than one duplicates (ie original baseline and
   % others)
```

```
   if length(duplicates{i, 2}) > 1
      % If the pair has not already been recorded
      if ~ismember(duplicates{i, 2}', usedPairs, 'rows')
           % Replace duplicate with average of both centroids
           baselineCentroids(duplicates{i, 1}, :) = (baselineCentroids(dupli-
cates{i, 1}, :) + centroidLocs{baselineFrames(1)}(duplicates{i, 2}(2), :)) / 2;
           % Delete duplicate centroid
           baselineCentroids(duplicates{i, 2}(2), :) = [];
           % Create composite pixel index with both centroids
           temp = false(512);
           temp(union(baselinePix{duplicates{i, 1}}, baselinePix{duplicates{i,
2}(2)})) = true;
           temp = imclose(temp, strel('disk', 3));
           baselinePix{duplicates{i, 1}} = find(temp);
           baselinePix{duplicates{i, 2}(2)} = [];
           usedPairs = [usedPairs; duplicates{i, 2}'];
       end
   end
end
baselinePix = baselinePix(~cellfun('isempty', baselinePix));

tempCent = cell(numFrames(dataNumber), 1);
tempFrameInd = cell(numFrames(dataNumber), 1);
% Delete empty cells
for k = baselineFrames
   tempCent{k} = [];
   tempFrameInd{k} = {};
   for j = 1 : length(newPixList{k})
      if ~isequal(centroidLocs{k}(j, :), [0 0])
         tempCent{k} = [tempCent{k}; centroidLocs{k}(j, :)];
         tempFrameInd{k}{end + 1} = newPixList{k}{j};
      end
   end
end

newPixList = tempFrameInd;
centroidLocs = tempCent;
for k = baselineFrames
    % Identify all of the centroids in the baseline frame
    for i = 1 : length(centroidLocs{k})
        % Check if centroid is already contained in baselinePix
        if any(ismembertol(centroidLocs{k}(i, :), baselineCentroids, 0.05, 'By-
Rows', true))
            [~, index] = ismembertol(centroidLocs{k}(i, :), baselineCentroids,
0.05, 'ByRows', true, 'OutputAllIndices', true);
             % Average current centroid with existing centroid
             baselineCentroids(index{1}(1), :) = (centroidLocs{k}(i, :) + base-
lineCentroids(index{1}(1), :)) / 2;
             % Store composite of both current and existing centroids
             temp = false(512);
             temp(baselinePix{index{1}(1)}) = true;
             temp(adj_pixList{k}{i}) = true;
             baselinePix{index{1}(1)} = find(temp);
        else % Centroid is not already in baselinePix

            % Store centroids and pixel index list
            baselineCentroids = [baselineCentroids; centroidLocs{k}(i, :)];
            baselinePix{end + 1} = adj_pixList{k}{i};
        end
    end
end
```

```
baselineCellAreas = zeros(length(baselineCentroids), 1);

for i = 1 : length(baselineCentroids)
   temp = false(512);
   temp(baselinePix{i}) = true;
   tempStats = regionprops(temp);
   baselineCellAreas(i) = tempStats.Area;
end
matchingCell = cell(numFrames(dataNumber), 1);

for k = baselineFrames
    % Store all centroids in the baseline frame
    cents = centroidLocs{k};
    matchingCell{k} = zeros(length(cents), 1);
    % Loop through all centroids in the baseline frame
    for i = 1 : length(cents)
       cellLoc = cents(i, :);
       dist = zeros(length(baselineCentroids), 1);
       % Loop through all baseline centroids
       for j = 1 : length(baselineCentroids)
          % Calculate distance from centroids in baseline frame to baseline
          % centroids
          dist(j) =  pdist([cellLoc; baselineCentroids(j, :)]);
       end
       % Find centroid with the minimum distance
       [~, I] = min(dist);
       % Match centroid in baseline frame to specific baseline centroid
       matchingCell{k}(i) = I;
    end
end
baselineFluor = zeros(length(baselineCentroids), length(baselineFrames));

for k = baselineFrames
   % Loop through all matched cells in baseline frames
   for i = 1 : length(matchingCell{k})
       % Store matched cell
       matchedCell = baselinePix{matchingCell{k}(i)};

       % Create mask outlining matched cell
       mask = false(512);
       mask(matchedCell) = true;

       % Multiply mask by original image and sum to get cell fluorescence
       tempIm = img_hrm63_raw{dataNumber}(:, :, k);
       indCell = tempIm .* uint8(mask);

       % Normalize cell to the baseline cell area
       indCellFluor = sum(sum(indCell)) / baselineCellAreas(match-
ingCell{k}(i));
       baselineFluor(matchingCell{k}(i), k - baselineFrames(1) + 1) = indCell-
Fluor;
   end
end


numBaselineCells = length(baselineCentroids);

keepInd = [];
```

```
for i = 1 : numBaselineCells
   if length(find(baselineFluor(i, :))) > 2
        keepInd = [keepInd; i];
   end
end
newBaselineFluor = baselineFluor(keepInd, :);
newBaselineCentroids = baselineCentroids(keepInd, :);
newBaselineCellAreas = baselineCellAreas(keepInd);
newBaselinePix = baselinePix(keepInd);

baselineFluorAvg = zeros(length(newBaselineFluor), 1);
for i = 1 : length(newBaselineFluor)
   baselineFluorAvg(i) = sum(newBaselineFluor(i, :)) / length(find(newBaseline-
Fluor(i, :)));
end
totalPix = cell(numFrames(dataNumber), 1);


totalFramesCentroids = cell(numFrames(dataNumber), 1);


totalFramesAreas = cell(numFrames(dataNumber), 1);


tempCent = cell(numFrames(dataNumber), 1);


tempFrameInd = cell(numFrames(dataNumber), 1);


for k = 1 : numFrames(dataNumber)
    disp(strcat('Reading Frame', {' '}, num2str(k)))

    frame1 = img_hrm63{dataNumber}(:, :, k);
    sub_mask = frame1;
    threshold = mean(mean(sub_mask));
    sub_mask(sub_mask >=  threshold) = threshold;
    sub_mask = imgaussfilt(sub_mask, 0.5);

    frame1 = frame1 - sub_mask;
    frame1 = imgaussfilt(frame1, 0.5);
    borders = zeros(512,512);

    dist1 = 1;
    threshUpper = 15;
    threshLower = 15;
    for r = 1 + dist1 : 512 - dist1
        for c = 1 + dist1 : 512 - dist1
            p    = frame1(r, c);
            pU1  = frame1(r - dist1, c);
            pL1  = frame1(r, c - dist1);
            pR1  = frame1(r, c + dist1);
            pB1  = frame1(r + dist1, c);

            AOI = [p pU1 pB1 pL1 pR1];

            if sum(AOI) ~= 0
                if p >= threshUpper
                    if ((pU1 <= threshLower && pB1 >= threshUpper) || (pU1 >=
threshUpper && pB1 <= threshLower))
                        borders(r, c) = 1;
                    elseif (pL1 <= threshLower && pR1 >= threshUpper) || (pL1
>= threshUpper && pR1 <= threshLower)
                        borders(r,c) = 1;
```

```
                    end
                end
            end
        end
    end

    bordersMem = bwmorph(borders, 'close');
    se = strel('disk',2);
    borders = imclose(bordersMem, se);

    for i = 1 : 512
       for j = 1 : 512
          if i == 512 || i == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif j == 512 || j == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif i == 512 && j == 512
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          elseif i == 0 && j == 1
              if frame1(i, j) >= 8
                borders(i, j) = 255;
              end
          end
       end
    end
    filled = imfill(borders,'holes');

    [labeledImage, numberOfRegions] = bwlabel(filled);

    labeledImage = bwareaopen(labeledImage, 100);

    binaryLabIm = zeros(512);
    binaryLabIm(labeledImage) = 1;
    borders(~labeledImage) = 0;

    filled = imfill(borders, 'holes');
    D = -bwdist(~filled);
    mask = imextendedmin(D, 2);
    D2 = imimposemin(D, mask);
    Ld = watershed(D2);
    filled(Ld == 0) = 0;
    L = filled;
    imshow(L)
    cc = bwconncomp(L, 4);
    tempPix = cc.PixelIdxList;
    stat = regionprops(cc, 'centroid', 'area');
    for i = 1 : length(stat)
         if stat(i).Centroid(1) < 510 && stat(i).Centroid(2) < 510 &&
stat(i).Centroid(1) > 10 && stat(i).Centroid(2) > 10
            if stat(i).Area > 10
               totalFramesCentroids{k} = [totalFramesCentroids{k}; stat(i).Cen-
troid];
               totalFramesAreas{k} = [totalFramesAreas{k}; stat(i).Area];
               totalPix{k}{1, i} = tempPix{i};
            end
```

```
        end
    end
    totalPix{k} = totalPix{k}(~cellfun('isempty', totalPix{k}));


   for j = 1 : length(totalPix{k})
      for i = 1 : length(totalPix{k})
         dist = pdist([totalFramesCentroids{k}(j, :); totalFramesCen-
troids{k}(i, :)]);
         cellArea1 = size(totalPix{k}{i}, 1);
         cellArea2 = size(totalPix{k}{j}, 1);
         if dist ~= 0 && dist < 40
             if cellArea1 < 400 || cellArea2 < 400
                 totalFramesCentroids{k}(j, :) = (totalFramesCentroids{k}(j, :)
+ totalFramesCentroids{k}(i, :)) / 2;
                 totalFramesCentroids{k}(i, :) = [0 0];
                 totalPix{k}{j} = [totalPix{k}{j}; totalPix{k}{i}];
                 totalPix{k}{i} = zeros(size(totalPix{k}{i}));
             end
         end
      end
   end

   tempCent{k} = [];
   tempFrameInd{k} = {};
   for j = 1 : length(totalPix{k})
      if ~isequal(totalFramesCentroids{k}(j, :), [0 0])
         tempCent{k} = [tempCent{k}; totalFramesCentroids{k}(j, :)];
         tempFrameInd{k}{end + 1} = totalPix{k}{j};
      end
   end

    totalPix = tempFrameInd;
    totalFramesCentroids = tempCent;

end

totalMatchingCell = cell(numFrames(dataNumber), 1);

for k = 1 : numFrames(dataNumber)
    cents = totalFramesCentroids{k};
    totalMatchingCell{k} = zeros(length(cents), 1);
    for i = 1 : length(cents)
       cellLoc = cents(i, :);
       dist = zeros(length(newBaselineCentroids), 1);
       for j = 1 : length( newBaselineCentroids)
          dist(j) = pdist([cellLoc; newBaselineCentroids(j, :)]);
       end
       [val, I] = min(dist);
       if val < 30
          totalMatchingCell{k}(i) = I;
       else
          totalMatchingCell{k}(i) = 0;
       end
    end
end

cellMasks = cell(numFrames(dataNumber), 1);
for k = 1 : numFrames(dataNumber)
   cellMasks{k} = cell(length(newBaselineCentroids), 1);
```

```
    for j = 1 : length(newBaselineCentroids)
        cellMasks{k}{j} = false(512);
    end
end

maskAreas = cell(numFrames(dataNumber), 1);
for k = 1 : numFrames(dataNumber)
    maskAreas{k} = zeros(length(newBaselineCentroids), 1);
    for j = 1 : length(totalMatchingCell{k})
        num = totalMatchingCell{k}(j);
        if num ~= 0
            temp = false(512);
            if ismember(0, totalPix{k}{j})
               totalPix{k}{j} = find(totalPix{k}{j});
            end
            temp(totalPix{k}{j}) = true;
            cellMasks{k}{num} = (cellMasks{k}{num} | temp);
            stats = regionprops(cellMasks{k}{num}, 'Area');
            maskAreas{k}(num) = stats.Area;
        end
    end
end
for k = 1 : numFrames(dataNumber)
    temp = false(512);
    for j = 1 : length(cellMasks{k})
       temp(cellMasks{k}{j}) = true;
    end
    imshow(imoverlay(adapthisteq(img_hrm63{dataNumber}(:, :, k)),
bwperim(temp)))
    pause(0.01)
end

totalCellFluor = zeros(length(newBaselineCentroids), numFrames(dataNumber));

for k = 1 : numFrames(dataNumber)
    for i = 1 : length(cellMasks{k})
        tempIm = img_hrm63_raw{dataNumber}(:, :, k);
        indCell = tempIm .* uint8(cellMasks{k}{i});
        indCellFluor = (sum(sum(indCell))) / maskAreas{k}(i);
        totalCellFluor(i, k) = indCellFluor;
    end
end
totalCellFluor(totalCellFluor == 0) = NaN;
norm_cell_fluor = zeros(size(totalCellFluor));

for k = 1 : numFrames(dataNumber)
    norm_cell_fluor(:, k) = (totalCellFluor(:, k) - baselineFluorAvg) ./ base-
lineFluorAvg;
end
norm_cell_fluor = norm_cell_fluor';
window = 2;

smooth_cell_fluor = zeros(numFrames(dataNumber) - window, length(newBaseline-
Centroids));

for i = 1 : length(newBaselineCentroids)
    for j = 1 : numFrames(dataNumber) - window
       smooth_cell_fluor(j, i) = nanmean(norm_cell_fluor(j : j + window, i));
    end
```

```
end
valid_cells = [];

for i = 1 : size(smooth_cell_fluor, 2)
   if sum(isnan(smooth_cell_fluor(:, i))) < 5 && max(smooth_cell_fluor(:, i)) <
1.8
       valid_cells = [valid_cells i];
   end
end
%treatmentTimes = [11 35.5 65.5 70.5 75.5];
treatmentTimes = [10.5 20.5 30.5 40.5];
for i = valid_cells
   fig1 = figure(1);
   plot(smooth_cell_fluor(:, i), 'o-')
   hold on
   for j = treatmentTimes
      line([j j], [0 20], 'color', 'r')
   end
   hold off
   ylim([0 2.5])
   xlim([0 51])
   fig1.Position = [100 480 560 420];

   fig2 = figure(2);
   temp = false(512);
   temp(newBaselinePix{i}) = true;
   figure(2)
   imshow(imoverlay(adapthisteq(img_hrm63{dataNumber}(:, :, base-
lineFrames(1))), bwperim(temp)))
   text(10, 10, num2str(i), 'color', 'r', 'FontSize', 12)
   fig2.Position = [700 262 686 607];
   pause(1)

end

plotCells = [1 : size(smooth_cell_fluor, 2)];

if window ~= 0
    padded_cell_fluor = [NaN(window, size(smooth_cell_fluor, 2));
smooth_cell_fluor];
else
    padded_cell_fluor = smooth_cell_fluor;
end
figure;
hold on
for i = 1 : length(treatmentTimes)
    line([treatmentTimes(i) treatmentTimes(i)], [-5 5], 'color', [50 * i 50 60
* i] / 255, 'linewidth', 2)
end
plot(padded_cell_fluor(:, valid_cells), 'o-')
ylim([-1 2.5])
title(strcat('Individual Cell Fluorescence of Hy46 with', {' '}, 'H_2', 'O_2',
{' '}, 'Ramp'), 'FontSize', 18)
legend('10 uM H202', '30 uM H202', '100 uM H202', '300 uM H202')
xlabel('Time (min)', 'FontSize', 14)
ylabel('\Delta F / F', 'FontSize', 14)
cellAreas = zeros(size(totalCellFluor));

for i = 1 : size(cellAreas, 1)
   for j = 1 : size(cellAreas, 2)
```

```
      cellAreas(i, j) = maskAreas{j}(i);
   end
end

cellAreas = cellAreas(valid_cells, :);
figure;


plot(nanmean(padded_cell_fluor(:, valid_cells), 2), 'linewidth', 2)
ylim([-1 2.5])
for i = 1 : length(treatmentTimes)
    line([treatmentTimes(i) treatmentTimes(i)], [-1 5], 'color', [50 * i 50 60
* i] / 255, 'linewidth', 2)
end
title(strcat('Average Across Cells of Hy46 with', {' '}, 'H_2', 'O_2', {' '},
'Ramp'), 'FontSize', 18)
legend('Average Signal', '10 uM H202', '30 uM H202', '100 uM H202', '300 uM
H202')
xlabel('Time (min)', 'FontSize', 14)
ylabel('\Delta F / F', 'FontSize', 14)

individual_cell_data{pos} = padded_cell_fluor(:, valid_cells);

figure;


leg = {};


for i = 1 : length(individual_cell_data)
   tempData = individual_cell_data{i};
   hold on
   plot(nanmean(tempData, 2), 'linewidth', 2.5)
   leg{end + 1} = cell2mat(strcat('Position', {' '}, num2str(i)));
end

leg{5} = '10 \muM (Pos 1/2 only)';
leg{6} = '30 \muM (Pos 1/2 only)';
leg{7} = '100 \muM (Pos 1/2 only)';
leg{8} = '300 \muM (All Pos)';

for i = 1 : length(treatmentTimes)
    line([treatmentTimes(i) treatmentTimes(i)], [-0.5 1.5], 'color', [50 * i 50
60 * i] / 255, 'linewidth', 2)
end
legend(leg)
xlabel('Time (min)', 'FontSize', 14)
ylabel('\Delta F / F', 'FontSize', 14)
title('Plate 2 HyPerRed Fluorescence', 'FontSize', 18)
```

7.2.4 Fiber Photometry Process Raw Tanks Code

```
clear all; close all; clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subjectName = 'nalfurafine_cage2_2TM_Subject1-200504-145923';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
```

```
DATAPATH = fullfile(MAINEXAMPLEPATH, 'Process raw files', 'Block Data'); %
\TDTMatlabSDK\Examples\ExampleData
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
BLOCKPATH = fullfile(DATAPATH, subjectName);

data = TDTbin2mat(BLOCKPATH, 'type', {'streams'});

sf = 1017;
sig_gcamp = data.streams.gcpA.data;
sig_isobe = data.streams.isoA.data;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ttl_pulse = data.epocs.PtC0.onset;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
timelin_gcamp = (1 : length(sig_gcamp)) / sf;

save (strcat(subjectName, '.mat'), 'sig_gcamp', 'sig_isobe','timelin_gcamp');%,
'ttl_pulse');

figure;
subplot(2,1,1)
plot(timelin_gcamp, sig_gcamp, 'k-')

subplot(2,1,2)
plot(timelin_gcamp, sig_isobe, 'k-')
```

7.2.5 Fiber Photometry Example Code

```
clear all; close all; clc
% Define sampling frequency, downsampling factor, and new sampling
% frequency
sf = 1017;
dsFactor = 100;
sf_ds = round(sf / dsFactor);

% Define paths for main folder, data, and images
folderPath = 'E:\Lab\Fiber Photometry\KLight';
dataPath = strcat(folderPath, '\KLight Data');
imagePath = strcat(folderPath, '\Images');

% Look in data path
cd(dataPath)

% Extract all files in data path
files = dir('**');
files = files(3 : end);

% Determine number of files
numFiles = length(files);

% Read in .mat files from data folder
data = cell(numFiles, 1);
for i = 1 : numFiles
   data{i} = open(files(i).name);
```

```
   disp(strcat('Reading File:', {' '}, files(i).name))
end

% Return to main folder path
cd(folderPath)

treatments = {'U50'; ...
              'Saline'; ...
              'Naloxone-U50'; ...
              'Morphine WD';...
              'Nalfurafine'};
numTreatments = length(treatments);
injections = cell(numFiles, 1);

treatmentInjections = [1,...
                       1,...
                       2,...
                       1,...
                       1];

% U50 Injection
injections{6} = 1124;
injections{7} = 682;
injections{8} = 974;
injections{9} = 1042;
injections{10} = 1040;

% Saline Injection
injections{1} = 997;
injections{2} = 1016;
injections{3} = 1020;
injections{4} = 1064;
injections{5} = 1094;

% Naloxone/U50 Injection
injections{11} = [1102 3004];
injections{12} = [1048 2990];
injections{13} = [1110 3026];
injections{14} = [1012 3008];
injections{15} = [1290 3234];

% Morphine W/D
injections{16} = 2004;
injections{17} = 1748;
injections{18} = 1886;
injections{19} = 1756;
injections{20} = 1834;

% Nalfurafine
injections{21} = 996;
injections{22} = 948;
injections{23} = 950;
injections{24} = 1342;
injections{25} = 1462;

numInjections = zeros(numFiles, 1);
for i = 1 : numFiles
   numInjections(i) = length(injections{i});
end

U50 = 6 : 10;
```

```
SALINE = 1 : 5;
NALOXONE_U50 = 11 : 15;
MORPHINE_WD = 16 : 20;
NALFURAFINE = 21 : 25;

% Define length of baseline in minutes
minute_baselineLength = 5;

% Convert baseline to seconds
baselineLength = minute_baselineLength * 60;

sig_klight = cell(numFiles, 1);


sig_isosbe = cell(numFiles, 1);




newInjections = cell(size(injections));

for i = 1 : numFiles
    newInjections{i} = injections{i} - injections{i}(1) + baselineLength;
end

% Truncate signal
for i = 1 : numFiles
    sig_klight{i} = downsample(data{i}.sig_gcamp, dsFactor);
    sig_isosbe{i} = downsample(data{i}.sig_isobe, dsFactor);

    % Cut off all signal before start of baseline period
    sig_klight{i} = sig_klight{i}((injections{i}(1) - baselineLength) * sf_ds :
end);
    sig_isosbe{i} = sig_isosbe{i}((injections{i}(1) - baselineLength) * sf_ds :
end);
end

sig_klight_norm = cell(numFiles, 1);


sig_isosbe_norm = cell(numFiles, 1);




sig_adj = cell(numFiles, 1);




bestk = zeros(numFiles, 1);




for i = 1 : numFiles
    % Normalize both KLight and Isosbestic channels
    sig_klight_norm{i} = ((sig_klight{i} - mean(sig_klight{i}(1 : base-
lineLength * sf_ds))) / mean(sig_klight{i}(1 : baselineLength * sf_ds))) * 100;
    sig_isosbe_norm{i} = ((sig_isosbe{i} - mean(sig_isosbe{i}(1 : base-
lineLength * sf_ds))) / mean(sig_isosbe{i}(1 : baselineLength * sf_ds))) * 100;
```

```
    % Low Pass Filter
    sig_klight_norm{i} = fft_LPF2(0.1, 0.3, sig_klight_norm{i}, sf_ds);
    sig_isosbe_norm{i} = fft_LPF2(0.1, 0.3, sig_isosbe_norm{i}, sf_ds);

    % Detrend
    sig_klight_norm{i} = detrend(sig_klight_norm{i});
    sig_isosbe_norm{i} = detrend(sig_isosbe_norm{i});

    % Scale isosbestic channel to fit KLight channel
    fun = @(x) sseval(x, sig_klight_norm{i}, sig_isosbe_norm{i});
    x0 = 0.5;
    bestk(i) = fminsearch(fun, x0);

    % Subtract fitted isosbestic channel from KLight channel
    sig_adj{i} = sig_klight_norm{i} - bestk(i) * sig_isosbe_norm{i};

end

clear sortByTreatment

% Create structure to contain data sorted by treatment
sortByTreatment(numTreatments) = struct();

% Initialize fields in sortByTreatment
for i = 1 : numTreatments
   sortByTreatment(i).treatment = treatments(i); % Contains treatment name
   sortByTreatment(i).rawData = {};              % Contains treatment data
   sortByTreatment(i).injectionTimes = {};       % Contains injection times
   sortByTreatment(i).id = {};                   % Contains mouse ID
end

% Initialize field for holding data for each injection period
for t = 1 : numTreatments
  for i = 1 : treatmentInjections(t)
      sortByTreatment(t).injections(i).injectionData = {};
  end
end

% Sort data into treatments
for i = 1 : numFiles
   % Determine treatment group
   if ismember(i, U50)
       tempTreat = 1;
   elseif ismember(i, SALINE)
       tempTreat = 2;
   elseif ismember(i, NALOXONE_U50)
       tempTreat = 3;
   elseif ismember(i, MORPHINE_WD)
       tempTreat = 4;
   elseif ismember(i, NALFURAFINE)
       tempTreat = 5;
   end
   % Add raw data to sortByTreatment
   sortByTreatment(tempTreat).rawData{end + 1, 1} = sig_adj{i}';

   % Add injection times to sortByTreatment
   sortByTreatment(tempTreat).injectionTimes{end + 1, 1} = newInjections{i};

   % Identify mouse ID
   tempID = regexp(files(i).name, '-', 'split');
   tempID = strcat(tempID{2}, '-', tempID{4}, '-', tempID{5}(1 : 2));
```

```
   % Add mouse ID to sortByTreatment
   sortByTreatment(tempTreat).id{end + 1, 1} = tempID;
end


for t = 1 : numTreatments % Go through each treatment group
  for i = 1 : treatmentInjections(t) % Go through each injection in each treat-
ment
      sortByTreatment(t).injections(i).minLength = 10 ^ 6;

      for rd = 1 : length(sortByTreatment(t).rawData) % Go through each mouse
         if treatmentInjections(t) == i % If the number of injections equals
the current injection of the treatment
            tempSig = sortByTreatment(t).rawData{rd}((sortByTreatment(t).injec-
tionTimes{rd}(i) - ...
                                                     sortByTreatment(t).injec-
tionTimes{rd}(1) + 1) * sf_ds : end);
         else
            tempSig = sortByTreatment(t).rawData{rd}((sortByTreatment(t).injec-
tionTimes{rd}(i) - ...
                                                     sortByTreatment(t).injec-
tionTimes{rd}(1) + 1) * sf_ds : ...
                                                     sortByTreatment(t).injec-
tionTimes{rd}(i + 1) * sf_ds);
         end
         % Store minimum length for each injection
         sortByTreatment(t).injections(i).minLength = min(length(tempSig),
sortByTreatment(t).injections(i).minLength);
         % Store injection data
         sortByTreatment(t).injections(i).injectionData{end + 1, 1} = tempSig;
      end
  end
end

multipleInjections = find(treatmentInjections > 1);
for t = multipleInjections
   for i = 1 : size(sortByTreatment(t).injections(2).injectionData , 1)
      sortByTreatment(t).injections(2).injectionData{i} = detrend(sortByTreat-
ment(t).injections(2).injectionData{i});
   end
end

exportDSFactor = 50;

exportSF = sf_ds / exportDSFactor;

smoothWindowSize = 1000;



for t = 1 : numTreatments
   for i = 1 : treatmentInjections(t)
      % Adjust minimum length to account for moving average window
      tempMin = sortByTreatment(t).injections(i).minLength - smoothWindowSize;

      % Initialize structure for holding post injection signals for all
      % mice
```

```
      sortByTreatment(t).injections(i).stackedInjection = ze-
ros(length(sortByTreatment(t).rawData), tempMin + smoothWindowSize);

      % Initialize structure for holding further downsampled export data
      sortByTreatment(t).injections(i).exportData = zeros(floor(tempMin / ex-
portDSFactor), length(sortByTreatment(t).rawData));
      sortByTreatment(t).injections(i).timeline = (1 : tempMin + smoothWin-
dowSize) / sf_ds - baselineLength;
      for rd = 1 : length(sortByTreatment(t).rawData)
         sortByTreatment(t).injections(i).stackedInjection(rd, :) =
sortByTreatment(t).injections(i).injectionData{rd}(1 : (tempMin + smoothWin-
dowSize));

         % Downsample and smooth signals
         tempExport = downsample(smoother(sortByTreatment(t).injec-
tions(i).stackedInjection(rd, :), smoothWindowSize), exportDSFactor);

         % Truncate tempExport to proper length
         tempExport = tempExport(1 : floor(tempMin / exportDSFactor));

         % Vertically shift signal to start at zero
         tempExport = tempExport - tempExport(60);
         %tempExport = tempExport - tempExport(1);

         sortByTreatment(t).injections(i).exportData(:, rd) = tempExport';
      end
      % Store timeline with exportData
      sortByTreatment(t).injections(i).exportData = [(1 : length(tempExport))'
/ exportSF - baselineLength sortByTreatment(t).injections(i).exportData];

      % Average each trace
      tempAvg = mean(sortByTreatment(t).injections(i).exportData(:, 2 : end),
2);

      % Store average with exportData
      sortByTreatment(t).injections(i).exportData = [sortByTreatment(t).injec-
tions(i).exportData tempAvg];
   end
end


cd(imagePath)

for t = 1 : numTreatments
   for i = 1 : treatmentInjections(t)
      for rd = 1 : length(sortByTreatment(t).rawData)
         fig = figure;
         plot(sortByTreatment(t).injections(i).timeline / 60, sortByTreat-
ment(t).injections(i).stackedInjection(rd, :))
         hold on
         line([0 0], [-10 20], 'color', 'r')
         xlim([-minute_baselineLength sortByTreatment(t).injections(i).time-
line(end) / 60])
         ylim([-10 20])
         title(strcat(sortByTreatment(t).treatment, {' '}, sortByTreat-
ment(t).id(rd), {' '}, 'Injection', {' '}, num2str(i)), 'FontSize', 18)
         ylabel('\Delta F / F', 'FontSize', 14)
         xlabel('Time (min)', 'FontSize', 14)
         legend('KLight Signal', 'Injection')
```

```
%           filetype = {'.jpg', '.fig'};
%           for f = 1 : length(filetype)
%               filename = char(strcat(sortByTreatment(t).treatment, '_',
sortByTreatment(t).id(rd), '_Injection_', num2str(i), filetype{f}));
%               saveas(fig, filename)
%           end
      end
   end
end
cd(folderPath)
fig = figure;

U50_leg = cell(size(sortByTreatment(1).injections.stackedInjection, 1), 1);

for i = 1 : size(sortByTreatment(1).injections.stackedInjection, 1)
    tempU50 = sortByTreatment(1).injections.exportData(:, 2 : end)';
    tempTime = sortByTreatment(1).injections.exportData(:, 1)';
    plot(tempTime / 60, tempU50(i, :))
    hold on
    U50_leg{i} = sortByTreatment(1).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 60])
ylim([-10 15])
legend(U50_leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual U50 Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

% saveFilename = 'Individual_U50_Traces.jpg';
% saveas(fig, saveFilename)

fig = figure;

sal_leg = cell(size(sortByTreatment(2).injections.stackedInjection, 1), 1);

for i = 1 : size(sortByTreatment(2).injections.stackedInjection, 1)
    tempsal = sortByTreatment(2).injections.exportData(:, 2 : end)';
    tempTimeSal = sortByTreatment(2).injections.exportData(:,1)';
    plot(tempTimeSal / 60, tempsal(i, :))
    hold on
    sal_leg{i} = sortByTreatment(2).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 60])
ylim([-5 12])
legend(sal_leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual Saline Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];
```

```
% saveFilename = 'Individual_Saline_Traces.jpg';
% saveas(fig, saveFilename)

fig = figure;


nal__leg = cell(size(sortByTreatment(3).injections(1).stackedInjection, 1), 1);

for i = 1 : size(sortByTreatment(3).injections(1).stackedInjection, 1)
    temp_nal = sortByTreatment(3).injections(1).exportData(:, 2 : end)';
    tempTimeNal = sortByTreatment(3).injections(1).exportData(:, 1)';
    plot(tempTimeNal / 60, temp_nal(i, :))
    hold on
    nal__leg{i} = sortByTreatment(3).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 35])
ylim([-10 25])
legend(nal__leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual Naloxone Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

% saveFilename = 'Individual_Naloxone_Traces.jpg';
% saveas(fig, saveFilename)

fig = figure;


nal__leg = cell(size(sortByTreatment(3).injections(2).stackedInjection, 1), 1);

for i = 1 : size(sortByTreatment(3).injections(2).stackedInjection, 1)
    temp_nal = sortByTreatment(3).injections(2).exportData(:, 2 : end)';
    tempTimeNal = sortByTreatment(3).injections(2).exportData(:, 1)';
    plot(tempTimeNal / 60, temp_nal(i, :))
    hold on
    nal__leg{i} = sortByTreatment(3).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 60])
ylim([-5 12])
legend(nal__leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual U50 post Naloxone Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

% saveFilename = 'Individual_U50_Post_Naloxone_Traces.jpg';
% saveas(fig, saveFilename)

fig = figure;


nal_wd__leg = cell(size(sortByTreatment(4).injections.stackedInjection, 1), 1);
```

```
for i = 1 : size(sortByTreatment(4).injections.stackedInjection, 1)
    temp_nal_wd = sortByTreatment(4).injections.exportData(:, 2 : end)';
    tempTimeNal_wd = sortByTreatment(4).injections.exportData(:, 1)';
    plot(tempTimeNal_wd / 60, temp_nal_wd(i, :))
    hold on
    nal_wd__leg{i} = sortByTreatment(4).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 60])
ylim([-5 12])
legend(nal_wd__leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual Naloxone Induced WD Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

% saveFilename = 'Individual_NaloxoneInducedWD_Traces.jpg';
% saveas(fig, saveFilename)

fig = figure;


nalf_leg = cell(size(sortByTreatment(5).injections.stackedInjection, 1), 1);

for i = 1 : size(sortByTreatment(5).injections.stackedInjection, 1)
    temp_nalf = sortByTreatment(5).injections.exportData(:, 2 : end)';
    tempTimeNalf = sortByTreatment(5).injections.exportData(:, 1)';
    plot(tempTimeNalf / 60, temp_nalf(i, :))
    hold on
    nalf_leg{i} = sortByTreatment(5).id{i};
end
line([0 0], [-10 25], 'color' ,'k', 'linewidth', 2)
xlim([-5 60])
ylim([-5 12])
legend(nalf_leg, 'location', 'bestoutside', 'box', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Individual Nalfurafine Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)

fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

% saveFilename = 'Individual_Nalfurafine_Traces.jpg';
% saveas(fig, saveFilename)


fig = figure;

tempsal_avg = sortByTreatment(2).injections.exportData(:, end)';



stdErr = std(sortByTreatment(2).injections.exportData(:, 2 : end - 1)') /
sqrt(size(sortByTreatment(2).injections.exportData(:, 2 : end - 1)', 1));
```

```
fillColor = [128 193 219] ./ 255;

lineColor = [ 52 148 186] ./ 255;

tempTimeSal = sortByTreatment(2).injections.exportData(:, 1)';

x_vector = [tempTimeSal / 60, fliplr(tempTimeSal / 60)];

patch = fill(x_vector, [tempsal_avg + stdErr, fliplr(tempsal_avg - stdErr)],
fillColor, 'handlevisibility', 'off');
set(patch, 'edgecolor', 'none');
set(patch, 'FaceAlpha', 0.4);
hold on
plot(tempTimeSal / 60, tempsal_avg, 'color', lineColor, 'LineWidth', 1);

tempU50_avg = sortByTreatment(1).injections.exportData(:, end)';
stdErr = std(sortByTreatment(1).injections.exportData(:, 2 : end - 1)') /
sqrt(size(sortByTreatment(1).injections.exportData(:, 2 : end - 1)', 1));
fillColor2 = [240 199 199] ./ 255;
lineColor2 = [ 245 76  76] ./ 255;
tempTime = sortByTreatment(1).injections.exportData(:, 1)';
x_vector = [tempTime / 60, fliplr(tempTime / 60)];
patch = fill(x_vector, [tempU50_avg + stdErr, fliplr(tempU50_avg - stdErr)],
fillColor2, 'handlevisibility', 'off');
set(patch, 'edgecolor', 'none');
set(patch, 'FaceAlpha', 0.4);
plot(tempTime / 60, tempU50_avg, 'color', lineColor2, 'LineWidth', 1);

tempNal_U50_avg = sortByTreatment(3).injections(2).exportData(:, end)';
stdErr = std(sortByTreatment(3).injections(2).exportData(:, 2 : end - 1)') /
sqrt(size(sortByTreatment(3).injections(2).exportData(:, 2 : end - 1)', 1));
fillColor2 = [120 196 159] ./ 255;
lineColor2 = [84 171 127] ./ 255;
tempTimeNal = sortByTreatment(3).injections(2).exportData(:, 1)';
x_vector = [tempTimeNal / 60, fliplr(tempTimeNal / 60)];
patch = fill(x_vector, [tempNal_U50_avg + stdErr, fliplr(tempNal_U50_avg -
stdErr)], fillColor2, 'handlevisibility', 'off');
set(patch, 'edgecolor', 'none');
set(patch, 'FaceAlpha', 0.4);
plot(tempTimeNal / 60, tempNal_U50_avg, 'color', lineColor2, 'LineWidth', 1);

tempNal_wd_avg = sortByTreatment(4).injections.exportData(:, end)';
stdErr = std(sortByTreatment(4).injections.exportData(:, 2 : end - 1)') /
sqrt(size(sortByTreatment(4).injections.exportData(:, 2 : end - 1)', 1));
fillColor3 = [221 169 245] ./ 255;
lineColor3 = [177 52 235] ./ 255;
tempTimeNal_wd = sortByTreatment(4).injections.exportData(:, 1)';
x_vector = [tempTimeNal_wd / 60, fliplr(tempTimeNal_wd / 60)];
patch = fill(x_vector, [tempNal_wd_avg + stdErr, fliplr(tempNal_wd_avg -
stdErr)], fillColor3, 'handlevisibility', 'off');
set(patch, 'edgecolor', 'none');
set(patch, 'FaceAlpha', 0.4);
plot(tempTimeNal_wd / 60, tempNal_wd_avg, 'color', lineColor3, 'LineWidth', 1);

tempNalf_avg = sortByTreatment(5).injections.exportData(:, end)';
stdErr = std(sortByTreatment(5).injections.exportData(:, 2 : end - 1)') /
sqrt(size(sortByTreatment(5).injections.exportData(:, 2 : end - 1)', 1));
```

```
fillColor4 = [232 211 162] ./ 255;
lineColor4 = [145 123 76] ./ 255;
tempTimeNalf = sortByTreatment(5).injections.exportData(:, 1)';
x_vector = [tempTimeNalf / 60, fliplr(tempTimeNalf / 60)];
patch = fill(x_vector, [tempNalf_avg + stdErr, fliplr(tempNalf_avg - stdErr)],
fillColor4, 'handlevisibility', 'off');
set(patch, 'edgecolor', 'none');
set(patch, 'FaceAlpha', 0.4);
plot(tempTimeNalf / 60, tempNalf_avg, 'color', lineColor4, 'LineWidth', 1);

ylim([-5 10])
xlim([-5 60])
line([0 0], [-5 15], 'color' ,'k', 'linewidth', 2, 'handlevisibility', 'off')
xlabel('Time (min)', 'FontSize', 26)
ylabel('\Delta F / F', 'FontSize', 26)
title('Average Treatment Traces', 'FontSize', 38)
set(gca, 'FontSize', 20)



fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 50 30];

legend({'Saline', 'U50', 'U50 Post Naloxone', 'Naloxone Induced WD', 'Nalfuraf-
ine'}, 'location', 'bestoutside', 'box', 'off')
saveFilename = 'Average_Treatment_Traces_Overlaid.jpg';

% legend({'Saline', 'Naloxone Induced WD'}, 'location', 'bestoutside', 'box',
'off')
% saveFilename = 'Saline_vs_NaloxoneWD_Overlaid.jpg';

%saveas(fig, saveFilename)
```

7.2.6 Delayed Alternation Code

```
clear all; close all; clc
% Set main folder path
folderPath = 'E:\Lab\Delayed Alternation\Flox KOR';
% Set data folder path
dataPath = strcat(folderPath, '\Flox KOR Data');
% Set image folder path
imagePath = strcat(folderPath, '\Images\Raster Plots');

cd(dataPath);
files = dir('**');
files = files(3 : end);
numFiles = length(files);

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

for filenum = 1 : numFiles
    filename = runs(filenum).filename;
    disp(strcat('Reading', {' '}, filename))
    loadin = regexp(fileread(filename), '\n', 'split').';
    splitl = regexp(loadin, '\s+', 'split');
    rawData= cell(length(splitl), 8);
    for i = 1 : length(splitl)
       D = splitl{i};
       for j = 1 : length(D)
```

```
            insert = D{j};
            if isempty(insert)
               insert = NaN;
            elseif ~isempty(str2num(insert))
               insert = str2num(insert);
            end
            rawData{i, j} = insert;
        end
    end
    runs(filenum).rawData = rawData;

    runs(filenum).datalocations.rowA = find(strcmp(rawData(:, 1), 'A:'));
    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:'));
    runs(filenum).datalocations.chamber = find(strcmp(rawData(:, 1), 'Box:'));

    count = 1;
    while (~isequal(splitl{count}{1}, 'Start'))
        count = count + 1;
    end
    dateTemp = splitl{count}{3};

    % Ensure correct date format of mm/dd/yy
    date = regexp(dateTemp, '/', 'split');
    dateCharMon = char(date(1));
    dateCharDay = char(date(2));
    if length(dateCharMon) == 1
        dateCharMon = strcat('0', dateCharMon);
    end
    if length(dateCharDay) == 1
       dateCharDay = strcat('0', dateCharDay);
    end
    date = strcat(date(3), dateCharMon, dateCharDay);
    runs(filenum).date = date;

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

   %Create A vector
    for boxnum = 1 : numBox
        Araw = rawData(runs(filenum).datalocations.rowA(boxnum) + 1 :
runs(filenum).datalocations.rowB(boxnum) - 1, 3 : 7);
        Atemp = [];
        for j = 1 : size(Araw, 1)
            current = Atemp;
            next5   = Araw(j, :);
            Atemp   = [current, next5];
        end
        runs(filenum).animal(boxnum).parseddata.A = Atemp;
    end

    %Create B vector
    for boxnum = 1 : numBox
        Braw = rawData(runs(filenum).datalocations.rowB(boxnum) + 1 :
runs(filenum).datalocations.rowC(boxnum) - 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(boxnum) - 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(boxnum) +
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

    for boxnum = 1 : numBox
        runs(filenum).animal(boxnum).chamberNum =  str2num(splitl{runs(file-
num).datalocations.chamber(boxnum)}{2});
    end
end
cd(folderPath);

for f = 1 : numFiles
    numBoxes = runs(f).numboxes;
    for ani = 1 : numBoxes
        if ~isempty(runs(f).animal(ani).parseddata.B)
            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
    end
end
flag = [];

for f = 1:numFiles
    if runs(f).numboxes > 2
```

```
        flag = [flag,f];
    else
        id = regexp(runs(f).filename(9:end),'\d+','match');
        if length(id) == 0;
            flag = [flag,f];
        else
        runs(f).animalID = str2num(id{1});
        end
    end
end

for i = 1:length(flag)
    a{i,1} = {runs(flag(i)).filename};
end
temp = {};

for i = 1 : numFiles
   temp{end + 1} = char(runs(i).date);
end
dates = unique(temp);

days = zeros(length(dates), 1);
for i = 1 : length(dates)
   wholeDate = str2num(dates{i});
   dd = mod(wholeDate, 100);
   wholeDate = floor(wholeDate / 100);
   mm = mod(wholeDate, 100);
   wholeDate = floor(wholeDate / 100);
   yy = wholeDate;
   days(i) = datenum(strcat(num2str(mm), '/', num2str(dd), '/', num2str(yy)),
'mm/dd/yy');
end
days = days - days(1) + 1;
days = days';

for i = 1 : length(dates)
    sortByDate(i).dates = dates{i};
end

numcorrect        = cell(length(dates),8,2);
nummissed         = cell(length(dates),8,2);
percentcorrect    = cell(length(dates),8,2);
totalresponses    = cell(length(dates),8,2);
rightpresses      = cell(length(dates),8,2);
leftpresses       = cell(length(dates),8,2);
correctaltsright  = cell(length(dates),8,2);
correctaltsleft   = cell(length(dates),8,2);
incorrectaltsright= cell(length(dates),8,2);
incorrectaltsleft = cell(length(dates),8,2);

for f = 1 : numFiles
    for d = 1 : length(dates)
        if char(runs(f).date) == dates{d}
            for id = 1 : 6
                if runs(f).animalID == id
                    for b = 1 : runs(f).numboxes

                        numcorrect{d,id,b}         = cell2mat(runs(f).ani-
mal(b).totals.correct_alts);
                        nummissed{d,id,b}          = cell2mat(runs(f).ani-
mal(b).totals.missed_alts);
```

```
                        percentcorrect{d,id,b}     = numcor-
rect{d,id,b}./(numcorrect{d,id,b}+nummissed{d,id,b});
                        totalresponses{d,id,b}     = numcorrect{d,id,b}+num-
missed{d,id,b};
                        rightpresses{d,id,b}       = cell2mat(runs(f).ani-
mal(b).totals.right_presses);
                        leftpresses{d,id,b}        = cell2mat(runs(f).ani-
mal(b).totals.left_presses);
                        correctaltsright{d,id,b}   = cell2mat(runs(f).ani-
mal(b).totals.correct_alts_right);
                        correctaltsleft{d,id,b}    = cell2mat(runs(f).ani-
mal(b).totals.correct_alts_left);
                        incorrectaltsright{d,id,b} = cell2mat(runs(f).ani-
mal(b).totals.incorrect_alts_right);
                        incorrectaltsleft{d,id,b}  = cell2mat(runs(f).ani-
mal(b).totals.incorrect_alts_left);

                        sortByDate(d).ID(id).box(b).numcorrect =
cell2mat(runs(f).animal(b).totals.correct_alts);
                        sortByDate(d).ID(id).box(b).nummissed  =
cell2mat(runs(f).animal(b).totals.missed_alts);
                        sortByDate(d).ID(id).box(b).percentcorrect = numcor-
rect{d,id,b}./(numcorrect{d,id,b}+nummissed{d,id,b});
                        sortByDate(d).ID(id).box(b).totalresponses = numcor-
rect{d,id,b}+nummissed{d,id,b};

                        sortByDate(d).ID(id).box(b).parsedData = runs(f).ani-
mal(b).parseddata;
                        sortByDate(d).ID(id).box(b).totals = runs(f).ani-
mal(b).totals;
                        chamberNum = runs(f).animal(b).chamberNum;
                        sortByDate(d).ID(id).box(b).chamberNum = chamberNum;
                    end
                end
            end
        end
    end
end

for f = 1 : length(sortByDate)
    for id = 1 : length(sortByDate(f).ID)
        numBoxes = length(sortByDate(f).ID(id).box);
        if numBoxes == 1
           if sortByDate(f).ID(id).box(1).chamberNum == 4
              fields = fieldnames(sortByDate(f).ID(id).box(1));
              tempStruct = cell(length(fields), 1);
              tempStruct = cell2struct(tempStruct, fields);
              sortByDate(f).ID(id).box = [tempStruct, sortBy-
Date(f).ID(id).box];
           end
        end
    end
end
for f = 1 : 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)
            if ~isempty(sortByDate(f).ID(id).box(b).parsedData)

               Emat = cell2mat(sortByDate(f).ID(id).box(b).parsedData.E);
               Tmat = cell2mat(sortByDate(f).ID(id).box(b).parsedData.T);
```

```
                Emat(isnan(Emat)) = [];
                Tmat(isnan(Tmat)) = [];

                right_lever_press_E = (Emat == 1);
                left_lever_press_E = (Emat == 2);
                correct_alts_right_E = (Emat == 15);
                correct_alts_left_E = (Emat == 16);
                incorrect_alts_right_E = (Emat == 17);
                incorrect_alts_left_E = (Emat == 18);
                delay_starts_E = (Emat == 23);
                delay_ends_E = (Emat == 24);
                incorrectAltTimes = Tmat(Emat == 17 | Emat == 18);
                correctAltTimes = Tmat(Emat == 15 | Emat == 16);
                totalAltTimes = sort([incorrectAltTimes correctAltTimes]);


                right_lever_press_T    = Tmat .* right_lever_press_E;
                left_lever_press_T     = Tmat .* left_lever_press_E;
                correct_alts_right_T   = Tmat .* correct_alts_right_E;
                correct_alts_left_T    = Tmat .* correct_alts_left_E;
                incorrect_alts_right_T = Tmat .* incorrect_alts_right_E;
                incorrect_alts_left_T  = Tmat .* incorrect_alts_left_E;
                delay_starts_T         = Tmat .* delay_starts_E;
                delay_ends_T           = Tmat .* delay_ends_E;


                right_lever_press_T(right_lever_press_T == 0 | is-
nan(right_lever_press_T)) = [];
                left_lever_press_T(left_lever_press_T == 0 | is-
nan(left_lever_press_T)) = [];
                correct_alts_right_T(correct_alts_right_T == 0 | isnan(cor-
rect_alts_right_T)) = [];
                correct_alts_left_T(correct_alts_left_T == 0 | isnan(cor-
rect_alts_left_T)) = [];
                incorrect_alts_right_T(incorrect_alts_right_T == 0 | isnan(in-
correct_alts_right_T)) = [];
                incorrect_alts_left_T(incorrect_alts_left_T == 0 | isnan(incor-
rect_alts_left_T)) = [];
                delay_starts_T(delay_starts_T == 0 | isnan(delay_starts_T)) =
[];
                delay_ends_T(delay_ends_T == 0 | isnan(delay_ends_T)) = [];

                sortByDate(f).ID(id).box(b).latencies.rlp =
right_lever_press_T;
                sortByDate(f).ID(id).box(b).latencies.llp = left_lever_press_T;
                sortByDate(f).ID(id).box(b).latencies.car = cor-
rect_alts_right_T;
                sortByDate(f).ID(id).box(b).latencies.cal = cor-
rect_alts_left_T;
                sortByDate(f).ID(id).box(b).latencies.iar = incor-
rect_alts_right_T;
                sortByDate(f).ID(id).box(b).latencies.ial = incor-
rect_alts_left_T;
                sortByDate(f).ID(id).box(b).latencies.dst = delay_starts_T;
                sortByDate(f).ID(id).box(b).latencies.den = delay_ends_T;
                sortByDate(f).ID(id).box(b).incorrectAltTimes = incor-
rectAltTimes;
                sortByDate(f).ID(id).box(b).correctAltTimes = correctAltTimes;
                sortByDate(f).ID(id).box(b).totalAltTimes = totalAltTimes;
```

```
                press_latency_binary_E = delay_ends_E + ...
                        correct_alts_right_E + correct_alts_left_E + ...
                        incorrect_alts_right_E + incorrect_alts_left_E;

                new_latency_binary_E = delay_starts_E + ...
                    correct_alts_right_E + correct_alts_left_E +...
                    incorrect_alts_right_E + incorrect_alts_left_E;

                press_latency_binary_T = Tmat .* press_latency_binary_E;
                new_latency_binary_T   = Tmat .* new_latency_binary_E;

                sortByDate(f).ID(id).box(b).latencies.plb = press_latency_bi-
nary_T;
                new_latency_binary_T(new_latency_binary_T == 0) = [];
                sortByDate(f).ID(id).box(b).latencies.nlb = new_latency_bi-
nary_T;

            end
         end
      end
   end
end
cd(imagePath)

for f = 1: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)
                if ~isempty(sortByDate(f).ID(id).box(b).latencies)

                figure;
                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-','lin-
ewidth',2); hold on;
                cl =
plot([cal;cal],[ones(1,length(cal)).*0;ones(1,length(cal))],'Col-
or',[1,.6,.2],'linewidth',2); hold on;

                ir =
plot([iar;iar],[ones(1,length(iar)).*0;ones(1,length(iar)).*-1],'b-','lin-
ewidth',2); hold on;
                il =
plot([ial;ial],[ones(1,length(ial)).*0;ones(1,length(ial)).*-1],'Col-
or',[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]);

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

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


                names = {'Incorrect {\bf\downarrow}';'Correct {\bf\uparrow}'};
                xlabel('Time (min)');
                ylim([-4,4])
                ax = gca;
                set(gca,'TickLength',[0,0])
                set(gca,'ytick',[-2,2],'yticklabel',names)
                ax.FontSize = 20;

                fig = gcf;
                set(gcf,'color','w');
                fig.PaperUnits = 'inches';
                fig.PaperPosition = [0 0 30 10];
                figureName = strcat('Flox KOR:', {' '}, sortBy-
Date(f).dates(1:end),{' '}, 'Chamber', {' '}, num2str(id),{' '}, 'Mouse', {'
'},num2str(b));
                title(figureName)

                text(0, 2, strcat(num2str(round(sortBy-
Date(f).ID(id).box(b).percentcorrect * 100, 3, 'significant')), '% Correct'),
'FontSize', 14)
                text(0, -2, strcat(num2str(100 - round(sortBy-
Date(f).ID(id).box(b).percentcorrect * 100, 3, 'significant')), '% Incorrect'),
'FontSize', 14)
%                 if ismember(sortByDate(f).dates, {'200318', '200319',
'200320'})
%                     saveas(gcf, strcat('Flox KOR_', sortBy-
Date(f).dates(1:end),'_',num2str(id),'_',num2str(b),'.jpg'))
%                 end
                end
            end
        end
    end
end
cd(folderPath)
numberOfBins = 2;

bins = (0 : 60 / numberOfBins : 60) * 60 * 100;



for f = 1 : 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)
                sortByDate(f).ID(id).box(b).binnedPC = [];
                sortByDate(f).ID(id).box(b).binnedTotal = [];
                sortByDate(f).ID(id).box(b).binnedCorrect = [];
         end
      end
   end
end


for ind = 2 : length(bins)
    for f = 1 : 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)
                  if ~isempty(sortByDate(f).ID(id).box(b).numcorrect)
                     % Total Alternation Times
                     totalAltTimes = sortByDate(f).ID(id).box(b).totalAltTimes;
                     totalCorTimes = sortByDate(f).ID(id).box(b).cor-
rectAltTimes;

                     temp_binTotal = length(totalAltTimes(totalAltTimes <=
bins(ind) & totalAltTimes > bins(ind - 1)));
                     temp_binCorrect = length(totalCorTimes(totalCorTimes <=
bins(ind) & totalCorTimes > bins(ind - 1)));
                     binPercentCorrect = temp_binCorrect / temp_binTotal;
                     sortByDate(f).ID(id).box(b).binnedPC = [sortBy-
Date(f).ID(id).box(b).binnedPC; binPercentCorrect];
                     sortByDate(f).ID(id).box(b).binnedTotal = [sortBy-
Date(f).ID(id).box(b).binnedTotal; temp_binTotal];
                     sortByDate(f).ID(id).box(b).binnedCorrect = [sortBy-
Date(f).ID(id).box(b).binnedCorrect; temp_binCorrect];
                  end
              end
           end
       end
    end
end

salDates = {'200102', '200103'};
u50Dates = {'200103', '200104'};

stackedBinPC_sal = [];
stackedBinTA_sal = [];
stackedBinCA_sal = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, salDates)
      if isequal(sortByDate(f).dates, salDates{2})
          for id = [2 3 5]
              if length(sortByDate(f).ID(id).box) >= 1
                  if id == 2 || id == 3
                      b = 1;
                  else
                      b = 2;
                  end
                  if ~isempty(sortByDate(f).ID(id).box(b).numcorrect)
                  stackedBinPC_sal = [stackedBinPC_sal; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                  stackedBinTA_sal = [stackedBinTA_sal; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
```

```
                    stackedBinCA_sal = [stackedBinCA_sal; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        end
    else
        for id = 1 : length(sortByDate(f).ID)
            for b = 1 : length(sortByDate(f).ID(id).box)
               if ~isempty(sortByDate(f).ID(id).box(b).numcorrect)
                    stackedBinPC_sal = [stackedBinPC_sal; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_sal = [stackedBinTA_sal; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_sal = [stackedBinCA_sal; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
               end
            end
        end
    end
end

stackedBinPC_u50 = [];
stackedBinTA_u50 = [];
stackedBinCA_u50 = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, u50Dates)
      if isequal(sortByDate(f).dates, u50Dates{1})
        for id = [2 3 5]
            if length(sortByDate(f).ID(id).box) >= 1
                if id == 2 || id == 3
                    b = 2;
                else
                    b = 1;
                end
                if ~isempty(sortByDate(f).ID(id).box(b).numcorrect)
                stackedBinPC_u50 = [stackedBinPC_u50; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                stackedBinTA_u50 = [stackedBinTA_u50; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                stackedBinCA_u50 = [stackedBinCA_u50; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        end
        for id = [1 4]
            for b = 1 : 2
                stackedBinPC_u50 = [stackedBinPC_u50; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                stackedBinTA_u50 = [stackedBinTA_u50; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                stackedBinCA_u50 = [stackedBinCA_u50; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
            end
        end
    else
        for id = 1 : length(sortByDate(f).ID)
            for b = 1 : length(sortByDate(f).ID(id).box)
               if ~isempty(sortByDate(f).ID(id).box(b).numcorrect)
                    stackedBinPC_u50 = [stackedBinPC_u50; sortBy-
Date(f).ID(id).box(b).binnedPC'];
```

```
                        stackedBinTA_u50 = [stackedBinTA_u50; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                        stackedBinCA_u50 = [stackedBinCA_u50; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                    end
                end
            end
        end
    end
end


stackedBinPC_sal = stackedBinPC_sal * 100;
stackedBinPC_u50 = stackedBinPC_u50 * 100;

sal_5mg_Dates = {'200109', '200110', '200111', '200122'};
u50_5mg_Dates = {'200110', '200111', '200112', '200123'};

stackedBinPC_sal_5mg = [];
stackedBinTA_sal_5mg = [];
stackedBinCA_sal_5mg = [];
for f = 1 : length(sortByDate)
    if ismember(sortByDate(f).dates, sal_5mg_Dates)
        if isequal(sortByDate(f).dates, sal_5mg_Dates{1})
            for id = [1 3 4 5]
                if length(sortByDate(f).ID(id).box) >= 1
                    if id == 1 || id == 5
                        for b = 2
                            stackedBinPC_sal_5mg = [stackedBinPC_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                            stackedBinTA_sal_5mg = [stackedBinTA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                            stackedBinCA_sal_5mg = [stackedBinCA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                        end
                    else
                        for b = 1 : 2
                            stackedBinPC_sal_5mg = [stackedBinPC_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                            stackedBinTA_sal_5mg = [stackedBinTA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                            stackedBinCA_sal_5mg = [stackedBinCA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                        end
                    end
                end
            end
        elseif isequal(sortByDate(f).dates, sal_5mg_Dates{2})
            for id = [2 5]
                for b = 1
                    stackedBinPC_sal_5mg = [stackedBinPC_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_sal_5mg = [stackedBinTA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_sal_5mg = [stackedBinCA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        elseif isequal(sortByDate(f).dates, sal_5mg_Dates{3})
            for id = [1 2]
               if id == 1
```

```
                b = 1;
            else
                b = 2;
            end
            stackedBinPC_sal_5mg = [stackedBinPC_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
            stackedBinTA_sal_5mg = [stackedBinTA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
            stackedBinCA_sal_5mg = [stackedBinCA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
        end
    else
        for id = 1 : length(sortByDate(f).ID)
            if length(sortByDate(f).ID(id).box) >= 1
                for b = 1 : length(sortByDate(f).ID(id).box)
                    stackedBinPC_sal_5mg = [stackedBinPC_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_sal_5mg = [stackedBinTA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_sal_5mg = [stackedBinCA_sal_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        end
    end
end

stackedBinPC_u50_5mg = [];
stackedBinTA_u50_5mg = [];
stackedBinCA_u50_5mg = [];
for f = 1 : length(sortByDate)
    if ismember(sortByDate(f).dates, u50_5mg_Dates)
        if isequal(sortByDate(f).dates, u50_5mg_Dates{1})
            if length(sortByDate(f).ID(id).box) >= 1
                for id = [1 3 4 5]
                    if id == 1 || id == 5
                        for b = 2
                            stackedBinPC_u50_5mg = [stackedBinPC_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                            stackedBinTA_u50_5mg = [stackedBinTA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                            stackedBinCA_u50_5mg = [stackedBinCA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                        end
                    else
                        for b = 1 : 2
                            stackedBinPC_u50_5mg = [stackedBinPC_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                            stackedBinTA_u50_5mg = [stackedBinTA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                            stackedBinCA_u50_5mg = [stackedBinCA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                        end
                    end
                end
            end
        elseif isequal(sortByDate(f).dates, u50_5mg_Dates{2})
            for id = [2 5]
                for b = 1
```

```
                    stackedBinPC_u50_5mg = [stackedBinPC_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_u50_5mg = [stackedBinTA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_u50_5mg = [stackedBinCA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        elseif isequal(sortByDate(f).dates, u50_5mg_Dates{3})
            for id = [1 2]
                if id == 1
                    b = 1;
                else
                    b = 2;
                end
                stackedBinPC_u50_5mg = [stackedBinPC_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                stackedBinTA_u50_5mg = [stackedBinTA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                stackedBinCA_u50_5mg = [stackedBinCA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
            end
        else
            for id = 1 : length(sortByDate(f).ID)
                if length(sortByDate(f).ID(id).box) >= 1
                    for b = 1 : length(sortByDate(f).ID(id).box)
                        stackedBinPC_u50_5mg = [stackedBinPC_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                        stackedBinTA_u50_5mg = [stackedBinTA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                        stackedBinCA_u50_5mg = [stackedBinCA_u50_5mg; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                    end
                end
            end
        end
    end
end


stackedBinPC_sal_5mg = stackedBinPC_sal_5mg * 100;
stackedBinPC_u50_5mg = stackedBinPC_u50_5mg * 100;



numSal_5mg = size(stackedBinPC_sal_5mg, 1);
numU50_5mg = size(stackedBinPC_u50_5mg, 1);
cd(folderPath)

sal_nal_Dates = {'200220'};
nal_Dates = {'200221'};

stackedBinPC_sal_nal = [];
stackedBinTA_sal_nal = [];
stackedBinCA_sal_nal = [];
for f = 1 : length(sortByDate)
    if ismember(sortByDate(f).dates, sal_nal_Dates)
        if isequal(sortByDate(f).dates, sal_nal_Dates{1})
            for id = 1 : length(sortByDate(f).ID)
                for b = 1 : length(sortByDate(f).ID(id).box)
```

```
                    stackedBinPC_sal_nal = [stackedBinPC_sal_nal; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_sal_nal = [stackedBinTA_sal_nal; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_sal_nal = [stackedBinCA_sal_nal; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        end
    end
end

stackedBinPC_nal = [];
stackedBinTA_nal = [];
stackedBinCA_nal = [];
for f = 1 : length(sortByDate)
    if ismember(sortByDate(f).dates, nal_Dates)
        if isequal(sortByDate(f).dates, nal_Dates{1})
            for id = 1 : length(sortByDate(f).ID)
                for b = 1 : length(sortByDate(f).ID(id).box)
                    stackedBinPC_nal = [stackedBinPC_nal; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                    stackedBinTA_nal = [stackedBinTA_nal; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                    stackedBinCA_nal = [stackedBinCA_nal; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                end
            end
        end
    end
end


stackedBinPC_sal_nal = stackedBinPC_sal_nal * 100;
stackedBinPC_nal = stackedBinPC_nal * 100;



numSal_nal = size(stackedBinPC_sal_nal, 1);
numNal = size(stackedBinPC_nal, 1);
cd(folderPath)

mor_Dates = {'200224', '200225', '200226', '200227', '200228'};
nox_Dates = {'200316', '200317', '200318', '200319', '200320'};

stackedBinPC_mor = [];
stackedBinTA_mor = [];
stackedBinCA_mor = [];
for f = 1 : length(sortByDate)
    if ismember(sortByDate(f).dates, mor_Dates)
        for id = 1 : length(sortByDate(f).ID)
            for b = 1 : length(sortByDate(f).ID(id).box)
                stackedBinPC_mor = [stackedBinPC_mor; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                stackedBinTA_mor = [stackedBinTA_mor; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                stackedBinCA_mor = [stackedBinCA_mor; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
            end
        end
    end
```

```
end

stackedBinPC_nox = [];
stackedBinTA_nox = [];
stackedBinCA_nox = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, nox_Dates)
      for id = 1 : length(sortByDate(f).ID)
          for b = 1 : length(sortByDate(f).ID(id).box)
              stackedBinPC_nox = [stackedBinPC_nox; sortBy-
Date(f).ID(id).box(b).binnedPC'];
              stackedBinTA_nox = [stackedBinTA_nox; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
              stackedBinCA_nox = [stackedBinCA_nox; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
          end
      end
   end
end


stackedBinPC_mor = stackedBinPC_mor * 100;
stackedBinPC_nox = stackedBinPC_nox * 100;



numMor = size(stackedBinPC_mor, 1);
numNox = size(stackedBinPC_nox, 1);
cd(folderPath)

veh_Dates = {'200130', '200131'};
sala_Dates = {'200131', '200202'};

stackedBinPC_veh = [];
stackedBinTA_veh = [];
stackedBinCA_veh = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, veh_Dates)
       if isequal(sortByDate(f).dates, veh_Dates{1})
          for id = [1 3 4]
              if id == 1 || id == 3
                  for b = 1 : length(sortByDate(f).ID(id).box)
                      stackedBinPC_veh = [stackedBinPC_veh; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                      stackedBinTA_veh = [stackedBinTA_veh; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                      stackedBinCA_veh = [stackedBinCA_veh; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                  end
              elseif id == 4
                  b = 2;
                  stackedBinPC_veh = [stackedBinPC_veh; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                  stackedBinTA_veh = [stackedBinTA_veh; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                  stackedBinCA_veh = [stackedBinCA_veh; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
              end
          end
       elseif isequal(sortByDate(f).dates, veh_Dates{2})
          id = 2;
```

```
          for b = 1 : length(sortByDate(f).ID(id).box)
              stackedBinPC_veh = [stackedBinPC_veh; sortBy-
Date(f).ID(id).box(b).binnedPC'];
              stackedBinTA_veh = [stackedBinTA_veh; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
              stackedBinCA_veh = [stackedBinCA_veh; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
          end
       end
   end
end

stackedBinPC_sala = [];
stackedBinTA_sala = [];
stackedBinCA_sala = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, sala_Dates)
      if isequal(sortByDate(f).dates, sala_Dates{1})
          for id = [1 3 4]
              if id == 1 || id == 3
                  for b = 1 : length(sortByDate(f).ID(id).box)
                      stackedBinPC_sala = [stackedBinPC_sala; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                      stackedBinTA_sala = [stackedBinTA_sala; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                      stackedBinCA_sala = [stackedBinCA_sala; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
                  end
              elseif id == 4
                  b = 2;
                  stackedBinPC_sala = [stackedBinPC_sala; sortBy-
Date(f).ID(id).box(b).binnedPC'];
                  stackedBinTA_sala = [stackedBinTA_sala; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
                  stackedBinCA_sala = [stackedBinCA_sala; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
              end
          end
      elseif isequal(sortByDate(f).dates, sala_Dates{2})
          id = 2;
          for b = 1 : length(sortByDate(f).ID(id).box)
              stackedBinPC_sala = [stackedBinPC_sala; sortBy-
Date(f).ID(id).box(b).binnedPC'];
              stackedBinTA_sala = [stackedBinTA_sala; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
              stackedBinCA_sala = [stackedBinCA_sala; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
          end

      end
   end
end


stackedBinPC_veh = stackedBinPC_veh * 100;
stackedBinPC_sala = stackedBinPC_sala * 100;



numVeh = size(stackedBinPC_veh, 1);
numSala = size(stackedBinPC_sala, 1);
```

```
cd(folderPath)

pres_Dates = {'181219', '190122'};
post_Dates = {'181220', '190123'};

stackedBinPC_pres = [];
stackedBinTA_pres = [];
stackedBinCA_pres = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, pres_Dates)
      for id = 1 : length(sortByDate(f).ID)
          for b = 1 : length(sortByDate(f).ID(id).box)
              stackedBinPC_pres = [stackedBinPC_pres; sortBy-
Date(f).ID(id).box(b).binnedPC'];
              stackedBinTA_pres = [stackedBinTA_pres; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
              stackedBinCA_pres = [stackedBinCA_pres; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
          end
      end
   end
end

stackedBinPC_post = [];
stackedBinTA_post = [];
stackedBinCA_post = [];
for f = 1 : length(sortByDate)
   if ismember(sortByDate(f).dates, post_Dates)
      for id = 1 : length(sortByDate(f).ID)
          for b = 1 : length(sortByDate(f).ID(id).box)
              stackedBinPC_post = [stackedBinPC_post; sortBy-
Date(f).ID(id).box(b).binnedPC'];
              stackedBinTA_post = [stackedBinTA_post; sortBy-
Date(f).ID(id).box(b).binnedTotal'];
              stackedBinCA_post = [stackedBinCA_post; sortBy-
Date(f).ID(id).box(b).binnedCorrect'];
          end
      end
   end
end


stackedBinPC_pres = stackedBinPC_pres * 100;
stackedBinPC_post = stackedBinPC_post * 100;



numPres = size(stackedBinPC_pres, 1);
numPost = size(stackedBinPC_post, 1);
cd(folderPath)
```