Ca^{2+}-dependent facilitation of $Ca_V2.1$ and flow cytometric FRET —a quantitative, model-based approach

by
Shin Rong Lee

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland

March, 2016

Abstract

This dissertation addresses two distinct questions through the common lens of a quantitative, biophysical approach. Both are informed by a model-based approach to biology, and enabled by cutting-edge experimental systems that permit the perturbation and simultaneous measurement of cellular signals.

The first part of this thesis describes how the brain-predominant voltage-gated Ca^{2+} channel ($Ca_V2.1$) is regulated by its permeant ion Ca^{2+}. By pairing novel electrophysiological techniques with Ca^{2+} imaging and uncaging, intracellular Ca^{2+} can be directly controlled through light, and the Ca^{2+}-dependent effects on channel gating resolved independently from confounding processes like ongoing voltage activation of channels. With this unprecedented control, we find surprisingly that Ca^{2+}-dependent facilitation of $Ca_V2.1$ is larger, faster and more Ca^{2+}-sensitive than previously imagined. These properties suggest that $Ca_V2.1$ furnishes exceptionally strong activity-dependent enhancement of Ca^{2+} entry throughout the nervous system, with important repercussions for downstream Ca^{2+}-dependent processes like neuronal plasticity.

The second part of this thesis moves to consider a high-throughput method for illuminating protein structure and function through fluorescence. It describes the development of a technique to measure with a flow cytometer single-cell protein levels and Förster resonance energy transfer (FRET) efficiencies in single live cells. With this flow cytometric FRET, we show

that binding curves can be rapidly constructed, and binding affinities obtained comparable to those using isothermal calorimetry. We applied this technique, with the aid of a FRET-based PKA activity sensor, to uncover the biochemical ramifications of a recently discovered Cushing's disease-causing mutant (L206R) of protein kinase A (PKA). We find that this point mutation within the catalytic domain of PKA not only differentially disrupted its ability to bind to select partners, but also caused an underlying catalytic deficiency. Our results provide mechanistic insight into the pathogenesis of L206R-mediated Cushing's syndrome, guiding potential future therapeutic strategies. They also showcase flow cytometric FRET as a rapid and convenient tool for assaying protein biochemistry within the highly relevant context of living cells.

Advisor: David T. Yue, M.D., Ph.D.
Readers: Lawrence Schramm, Ph.D. and Dwight Bergles, Ph.D.

Acknowledgements

I have accumulated much debt during the course of my dissertation research. With thanks to the Hopkins Medical Scientist Training Program and the NIH that this debt is not a financial one.

I am immensely grateful for my advisor, Dr. David Yue, who was a dedicated mentor and a brilliant scientist. His exuberance for teaching and research was the tide that raised all boats in the lab. He taught through example that good, rigorous science is careful and deliberate, instilling within my rash heart a deep well of patience. He challenged my pessimism, and showed me the surprising joys that optimism can bring. He is sorely missed, but his teachings, wisdom and work will ever remain a part of me.

I am also very thankful for Dr. Dwight Bergles, who not only contributed valuably to my dissertation through his work in my thesis committee, but who also kindly and diligently helped to fill the void left upon David's passing. It was only through Dr. Bergles' generous support that I was able to apply to residency programs in 2015, and for that I'm eternally grateful.

Much appreciation and thanks also to my other committee members, Dr. Larry Schramm, Dr. Frank Bosmans, and Dr. Linggang Wu, who all perilously agreed to undertake this journey with me, providing essential feedback along the way.

To the MSTP program, our fearless leaders Dr. Bob Siliciano and Dr. Andrea Cox, Sharon Welling, Bernadine Harper and Martha Buntin, thank you all for your support and encouragement throughout these years. Thank you also to the BME department, including Dr. Elliot McVeigh, Dr. Reza Shadmehr, Hong Lan, Regina Tan, Deon Grant, and Joyce Bankert for fostering a nurturing environment for all budding engineer-scientists.

To the Calcium Signals Lab, whose members are as a second family to me, a heartfelt thanks to all of you. Through joy and tears we have toiled and lived, and this dissertation is truly the result of our combined efforts. Special thanks to Dr. Paul Adams, who introduced me to the world of neurophysiological research by teaching me everything from animal husbandry to recording electrical and fluorescent signals from brain slices. Without his help, part 1 of this thesis would not have been possible. Thanks also to Lingjie Sang, who worked with me to establish flow cytometric FRET as a viable method for performing protein biochemistry. Without her aid, part 2 of this thesis could not have been completed. I'm indebted to Dr. Manu ben Johny, Dr. Ivy Dick, and Wanjun Yang, each of whom were instrumental in teaching me the core skills of electrophysiology, fluorescence imaging and molecular biology. Thank you also to Dr. Nancy Yue, for sharing David with the lab, and whose strength carried us through the most difficult of times.

And finally, my deepest thanks to my family. To my wife, Shu-han Luo, who encourages me to pursue my dreams even when it conflicts with her own and is supportive beyond reason, my brother, Shin Yin Lee, who provides necessary auxiliary support at my beck and call, my in-laws, Tingyao Luo and Yahuei Lin, who model academic excellence and spoil me with gifts and affection, and my parents, King Sim Lee and Tiou Cheng Liang, who selflessly devote themselves to me, and to whom I owe absolutely everything.

Table of Contents

Abstract ii
Acknowledgements iv
Table of Contents vii
List of Figures ix
List of Tables ix
Part 1: Ca^{2+}-dependence of $Ca_V2.1$ 1
Introduction 2
Methods 6
Molecular Biology 6
Cell culture and transfection 7
Dissociated Purkinje Neurons 7
Electrophysiology 8
Whole-cell solutions 8
Ca^{2+} imaging and uncaging 9
Simulations – $Ca_V2.1$ channel models 10
Physiological model 11
Results 12
Minimizing Ca^{2+} block of Li^+ currents carried by $Ca_V2.1$ channels 12
Steady-state Ca^{2+} sensitivity of $Ca_V2.1$ CaM regulation revealed by Ca^{2+} uncaging 14
Contrasts between fully facilitated and non-facilitated channels at various voltages 17
Voltage insensitivity of Ca^{2+}-driven conversion to facilitated channels 21
Dynamic Ca^{2+} responsiveness of CDF 22
Devising strategies for resolving native $Ca_V2.1$ properties 24
Prominent CDF in native CaV2.1 in cerebellar Purkinje neurons 28
Discussion 30
Structural determinants of CDF 30
Advances in functional mechanisms 31
Neurophysiological Implications 33
Figures 38

Part 2: Uncovering aberrant mutant PKA function with flow cytometric FRET 56
Introduction 57
Methods 60
Molecular biology 60
Cell culture, transfection and preparation for flow cytometry 61
Flow cytometry 62
Data processing 63
Results 64
Quantitative FRET with flow cytometry 64
Absolute calibration of the flow cytometer 66
Live cell FRET-based binding curves 68
Regulation of Protein Kinase A—implications of a catalytic subunit mutant (L206R) 70
Discussion 75
Strengths and Limitations of Flow Cytometric FRET 76
Implications of L206R for PKA function 77
Figures and Tables 79
Appendix 92
IA. Simulation code for 2 compartment model 92
IIA. Quantitative FRET with flow cytometry 95
IIB. Binding model for PKA_{reg} and PKA_{cat} 96
IIC. AKAR4 activity with PKA_{cat} 96
IID. AKAR4 activity in presence of an inhibitor 97
IIE. Matlab code for flow cytometric FRET 98
References 146
Curriculum Vitae 153

List of Figures

Figure 1. Ca^{2+} sources influencing $Ca_V2.1$ opening 38
Figure 2. Relieving Ca^{2+} block of Li^+ current 39
Figure 3. Steady-state Ca^{2+} sensitivity of $Ca_V2.1$ revealed by Ca^{2+} uncaging 40
Figure 4. $Ca_V2.1$ EFb +47 lacks CDF 42
Figure 5. High Ca^{2+} requirements for CDI 43
Figure 6. Contrasts between facilitated and non-facilitated channels at various potentials 44
Figure 7. Physiologically-relevant action potential waveforms have significant CDF 46
Figure 8. Voltage-insensitivity of Ca^{2+}-driven conversion to facilitated channels 47
Figure 9. Dynamic Ca^{2+} responsiveness of CDF 48
Figure 10. Towards resolving native $Ca_V2.1$ CDF 50
Figure 11. CDF in native $Ca_V2.1$ channels within cerebellar Purkinje cells 52
Figure 12. Neurophysiological implications of $Ca_V2.1$ CDF 54
Figure 13. Flow cytometric FRET 80
Figure 14. Selection, background fluorescence, and size of single cells 81
Figure 15. Accounting for cross-talk across channels 82
Figure 16. Calibrating the flow cytometer for FRET 84
Figure 17. Effects of immature FPs on FRET quantitation 85
Figure 18. Flow cytometric FRET binding curves 86
Figure 19. Determining K_d and E_{max} 87
Figure 20. Differential effects on binding by PKA_{cat} L206R 88
Figure 21. Further structure-function studies of PKA 89
Figure 22. Enzyme kinetics of PKA L206R revealed through a FRET-based PKA sensor 90

List of Tables

Table 1. List of primers used 79

Part 1: Ca^{2+}-dependence of $Ca_V2.1$

Introduction

$Ca_V2.1$ channels are perhaps the most abundant voltage-gated Ca^{2+} channel in the mammalian brain [1], furnishing a dominant Ca^{2+} entry pathway into presynaptic release terminals [2-5], as well as into somatic and postsynaptic compartments [6-8]. These channels thereby constitute the pre-eminent trigger of neurotransmitter release in the CNS [9], initiate Ca^{2+}-dependent forms of synaptic and developmental plasticity [10-14], and support and modulate dendritic Ca^{2+} spiking important for motor coordination [15, 16]. In all, dynamic regulation of $Ca_V2.1$ activity is likely a significant contributor to the complexity and richness of neurophysiology and neurocomputation.

Early studies in squid reported that the amplitude of voltage-gated Ca^{2+} currents changed little with altered stimulation patterns, suggesting that other Ca^{2+}-dependent processes were responsible for differing forms of rapid Ca^{2+}-dependent plasticity [17]. For example, paired-pulse facilitation of synaptic efficacy has been attributed to the accumulation of residual Ca^{2+} that primes Ca^{2+} triggering sites for vesicle release, and short-term synaptic depression has been explained in terms of vesicle depletion [18]. These mechanisms have attracted considerable interest, because short-term synaptic plasticity is considered to be essential for the neurocomputational repertoire of the brain [19]. That said, if the amplitude of $Ca_V2.1$ currents were variably modulated by recent activity, an additional and powerful class of plasticity mechanisms would arise.

More recently, rapid voltage-dependent inactivation (VDI) has been identified in $Ca_V2.1$ channels paired with certain auxiliary beta subunits [20]. In certain contexts where such subunits predominate, it has been argued that such VDI could underlie forms

of short-term synaptic depression [21]. Apart from VDI, Ca^{2+}/calmodulin (Ca^{2+}-CaM) regulation of Cav2.1 channels has emerged as another potential mechanism supporting activity-dependent plasticity [22, 23]. Intriguingly, CaM regulation of Cav2.1 exhibits a bipartite format relating to the lobes of an indwelling CaM bound to the intracellular carboxy tail of channels [22, 24]. Ca^{2+} binding to the C-terminal lobe of CaM induces CDF, whereas Ca^{2+} binding to the N-terminal lobe of CaM induces a Ca^{2+}-dependent inactivation (CDI) [22]. Indeed, there have been hints that such forms of regulation are at play within neuronal preparations [25-29], and there have been significant efforts to clarify the structure-function underpinnings of such channel regulation [22, 30-32].

Nonetheless, progress has been stymied by the inability to quantify CDF and CDI in more than a cursory qualitative manner. Even the coarse Ca^{2+} sensitivity and magnitude of these forms of regulation remain obscure, complicating efforts to ascribe physiological relevance and decipher structure-function mechanisms. These limitations are particularly acute for CDI of Cav2.1 channels, which can only be robustly induced when large Ca^{2+} currents are present with minimal levels of Ca^{2+} buffering [33]. Thus, the levels of Ca^{2+} required for CDI may exceed physiological bounds.

To overcome these limitations, we scrutinized the obstacles to determining the Ca^{2+} sensitivity of Cav2.1 regulation as diagrammed in Figure 1*A*. One typically quantifies channel regulation by measuring whole-cell Ca^{2+} currents (*I*), whose magnitude is specified by channel activation gating (top left box) that is modulated by a CaM regulatory system (top right box in red). However, the Ca^{2+} concentration at the inner mouth of the channel, labeled as nanodomain Ca^{2+} in Figure 1*B*, constitutes the actual

input to the CaM regulatory module, and this nanodomain Ca^{2+} is difficult to measure and control. The challenges arise because nanodomain Ca^{2+} results from the superposition of local Ca^{2+} fluxing through a 'home' channel (Figure 1*B*, red arrow) and global Ca^{2+} sources comprised of other Ca^{2+} channels, transport pathways, and internal stores. The local Ca^{2+} input is complex, reflecting stochastic channel gating, as well as local diffusion and Ca^{2+} buffering near the channel mouth (Figure 1*A*, 'local' pathway). The global Ca^{2+} input derives from the activity of other Ca^{2+} channels and sources at a distance, as shaped by cellular Ca^{2+} diffusion, geometry, and buffering (Figure 1*A*, 'global' pathway). There are numerous unknowns among these multiple determinants of local and global Ca^{2+} inputs. Moreover, the relevant nanodomain Ca^{2+} concentration cannot be measured in most instances, because commonly used Ca^{2+} fluorescent dyes only furnish readouts of spatially averaged global Ca^{2+} (Figure 1*B*, light-shaded region), and more intricate near-field measurements are presently impractical for the present question [34]. For CDF, further challenges arise because channel facilitation transpires rapidly over a timescale similar to the growing opening of channels following the onset of a voltage step (Figure 1*A*, 'activation gating'). Hence, it is difficult to disambiguate the contributions of CaM-mediated CDF and voltage activation. As a result, prior attempts to quantify CDF have done so indirectly, either by estimating the degree to which channels can be pre-facilitated during a preceding voltage step [22, 25, 35], or by monitoring the gradual increase in currents in response to a train of brief voltage steps [23, 30, 35]. Though useful for appraising relative changes in CDF, such methods do not quantify the

absolute extent of CDF, and can only assess regulation over a narrow voltage range near the midpoint of voltage activation.

Thus appraised, we here devised strategies to directly measure and control nanodomain Ca^{2+} while recording $Ca_V2.1$ whole-cell currents at steady state with respect to activation gating. Figure 1*C* and **Figure *1D*** schematize the overall approach, where Li^+ is substituted for Ca^{2+} as charge carrier through these channels (Figure 1*D*). In so doing, Ca^{2+} influx through channels no longer triggers CaM regulation, and channel gating is uncoupled from induction of the CaM regulatory module (Figure 1*C*). Instead, uniformly dialyzed DM-nitrophen (caged Ca^{2+}) is photolyzed to abruptly produce spatially uniform Ca^{2+} elevations throughout the cytoplasmic compartment, including the channel nanodomain (Figure 1*D*). Furthermore, ratiometric Ca^{2+} fluorescent dyes permit real-time measurement of Ca^{2+} concentration. Accordingly, we directly control and measure the Ca^{2+} input to the CaM regulatory module (Figure 1*C*) [36, 37]. Importantly, photouncaging can be produced well after channels have achieved steady state with respect to voltage activation, so that the changes in $Ca_V2.1$ current directly reflect the effects of CaM regulation alone (Figure 1*C*, steady gating).

Using this approach, we now find that CDF of $Ca_V2.1$ is larger, faster, and more Ca^{2+} sensitive than expected. Importantly, the properties of CDF render Ca^{2+} current modulation particularly strong during physiological spike-like voltage stimuli. By contrast, CDI turns out to be more than 50-fold less Ca^{2+} sensitive, such that CDF can in many instances develop in isolation from CDI. Our results pertain to both recombinant $Ca_V2.1$ channels and native $Ca_V2.1$ currents within cerebellar Purkinje neurons.

Moreover, $Ca_V2.1$ behavior in both contexts could be quantitatively predicted by a common biophysical model of $Ca_V2.1$ gating and CDF, gleaned from extensive experimental constraints. Overall, CDF of $Ca_V2.1$ channels is well positioned to influence activity-dependent processes throughout the CNS.

Methods

Molecular Biology

The parental human $Ca_V2.1$ clones were gifts from Dr. Terry Snutch (University of British Columbia). The exact splice background [33] used was: Δ10A (+G); $16^+/17^+$; Δ17A (-VEA); -31* (-NP); 37a (EFa) or 37b (EFb); $43^+/44^+$; 47^- or 47^+. Henceforth, we will for compactness refer to the main constructs as EFa -47, EFb +47, and EFb -47.The entire channel ORF was transferred into the pEGFP-N3 vector (Clontech) with restriction enzymes Nhe1/XbaI for ease of mutagenesis (E-GFP coding sequence replaced). The EFb variant in this same background was generated by transferring the relevant region from our $Ca_V2.1$ EFb clone [33] with XhoI/BglII. The pore mutants were all generated by overlap-extension PCR, as previously described [38]. The primers used to generate E1A were: CCCGTCCGACAGAACTGCCTC (flank1_F), GATCAGTCCACCCTgCCATGGTTATGC (E1A_R), GCATAACCATGGcAGGGTGGACTGATC (E1A_F), and TAGGGTCCCTCCCGGCTCAG (flank1_R). The PCR product containing the mutation was then transferred back into our channel backbone with AgeI/PvuI. E2A was generated with the same flanking primers and restriction enzymes, with these overlap primers: CCTGACGGGCGcAGACTGGAAC (E2A_F) and GTTCCAGTCTgCGCCCGTCAGG (E2A_R). The primers for E3A were AACAGGCCCGCTACCACG (flank3_F),

CCAGCCTgCTCCCGTGGAC (E3A_R), GTCCACGGGAGcAGGCTGG (E3A_F), and GAAGTTATTCCCAAACTCAGTCACGAGG (flank3_R), and transferred with enzymes PvuI/EcoRV. The primers for E4A were TGTTCTGGGCAGCATCACCG (flank4_F), GCCAAGCTgCCCCGGTGG (E4A_R), CCACCGGGGcAGCTTGGC (E4A_F), and CAGCTTCTTGGCCTTGCTCTGC (flank4_R), and inserted with enzymes EcoRV/BglII. All mutants were verified by Sanger sequencing.

Cell culture and transfection

HEK293 cells were sterilely cultured on glass coverslips in 60mm plates at 37ºC with 5% CO_2, and transiently transfected with a calcium phosphate protocol [39]. We used 2-5 μg of cDNA encoding the desired channel α_1 subunit, with 4 μg each of rat brain β2a and α2δ subunits. 1 μg of SV40 T-antigen and 1 μg of CFP were also co-transfected to enhance expression and identify transfected cells. For experiments not involving Ca^{2+}-uncaging, β2a-IRES2-eGFP was used in place of β2a plus CFP to identify transfected cells. All constructs were driven by a CMV promoter, and cells were used for electrophysiology 1-3 days after transfection.

Dissociated Purkinje Neurons

Cerebellar Purkinje neurons were isolated from postnatal day 14-18 C57/B6 mice (Jackson Labs) as previously described [25]. Animals were handled according to protocols approved by the Johns Hopkins University's Animal Care and Use Committee. Briefly, Purkinje neurons were acutely dissociated from cerebellar cortex and plated on poly-D-lysine coated glass-bottom dishes (In Vitro Scientific). Cells were allowed to recover for an hour at room temperature before experimentation.

Electrophysiology

For both HEK293 cells and Purkinje neurons, whole-cell recordings were obtained using Axopatch 200 or 200A amplifiers (Axon Instruments) at room temperature. Electrodes were pulled (model P-97, Sutter Instruments) from borosilicate glass capillaries (World Precision Instruments, MTW 150-F4) and fire-polished with a microforge (Narishige) to achieve resistances of 1-3 MΩ before series resistance compensation of 70-80%. Currents were low-pass filtered at 5kHz, and leak-subtracted with a P/4-P/8 protocol. Data acquisition (digitized with ITC-18 (Instrutech)) and analysis of electrophysiological data were done with custom software written in Matlab (MathWorks).

Whole-cell solutions

Internal solutions: For the Ca^{2+} block and pre-pulse CDF characterizations, the internal solution contained (in mM): 135 $CsMeSO_3$, 5 CsCl, 1 $MgCl_2$, 4 MgATP, 1 EGTA, 10 HEPES with pH titrated to 7.4 with CsOH, and adjusted to 295 mOsm with glucose. For Ca^{2+}-uncaging experiments, the internal solution contained (in mM): 135 CsCl, 40 HEPES (pH 7.4), 1-2 Citrate, 5 μM Fluo-2 (TefLabs) + 5 μM Fluo-2 Low Affinity (TefLabs), 2.5 μM Alexa568 (Invitrogen), 2 DMNP-EDTA (Invitrogen) and 0.25-1 $CaCl_2$. The two Ca^{2+} dyes with different Ca^{2+} sensitivities were combined with Alexa568 for increased dynamic range of ratiometric Ca^{2+} detection. Calibration of premixed stocks of Fluo/Alexa was performed both *in vitro* with droplets and also *in vivo* by patching HEK293 cells in the whole-cell configuration with internal solutions containing different free Ca^{2+}, buffered by either 5 mM EGTA, 5 mM HEDTA, or 5 mM NTA. Free Ca^{2+} concentrations were calculated with MaxChelator (Stanford).

External solutions: For characterizations of CDF with the pre-pulse protocol, the bath solution contained (in mM): 140 TEA-$MeSO_3$, 10 HEPES (pH 7.4 with TEA-OH), and 5 $CaCl_2$ or 5 $BaCl_2$ adjusted to 295 mOsm with glucose. For Ca^{2+} block experiments, bath solution contained (in mM): 80 TEA-$MeSO_3$, 10 HEPES (pH 7.4), 80 LiCl, and 0.5-2.5 $CaCl_2$ buffered by 5 EGTA, 5 HEDTA or 5 NTA depending on the desired concentration of free Ca^{2+} (295 mOsm with glucose). Free Ca^{2+} concentrations of these solutions were calculated using MaxChelator (Stanford). For Ca^{2+}-uncaging experiments, bath solution contained (in mM): 80 TEA-$MeSO_3$, 10 HEPES (pH 7.4), 80 LiCl, and 2 EGTA (295 mOsm with glucose). Unless otherwise specified, all reagents were obtained from Sigma Aldrich.

Ca^{2+} imaging and uncaging

All Ca^{2+}-uncaging experiments were done on a Nikon TE2000-U inverted microscope with a 40x/1.3 Plan Fluor objective as previously described [36]. Briefly, dyes are excited by a 514nm Argon laser, with the resulting fluorescence obtained at 2 channels after filtering by a 545DCLP dichroic, and either 545/40BP (Fluo) or 580LP (Alexa568). Brief UV pulses were generated by a Cairn UV flash photolysis system, with varying UV intensities achieved by adjusting the voltage (50-200V) and total capacitance (500-4000μF) of the capacitor bank. The fluorescence of our 2 dyes as a function of Ca^{2+} is a superposition of the Ca^{2+} response of each dye, and exhibits the following relation:

$$R = \left(R_{min,F2} + \frac{\left(R_{max,F2} - R_{min,F2}\right)\cdot Ca}{K_{D,F2} + Ca} \right) + \left(R_{min,F2LA} + \frac{\left(R_{max,F2LA} - R_{min,F2LA}\right)\cdot Ca}{K_{D,F2LA} + Ca} \right)$$

where R is the measured green/red fluorescence ratio. We determined R_{min} and R_{max} by whole-cell dialysis of HEK293 cells with carefully buffered Ca^{2+} solutions, and verified that the *in vivo* K_d values for both dyes (0.39 and 6.7 μM) were the same as the *in vitro* K_d values provided by the manufacturer (Teflabs). Intracellular Ca^{2+} was then determined from the measured R by numerically inverting the above equation.

Simulations – $Ca_V2.1$ channel models

The basic unfacilitated channel model consists of 6 states, with 4 transitions amongst 5 closed states before a final transition to the open state. The parameters of these rate constants were chosen such that they fit both the steady-state P_o versus voltage relations (Figure 6*E*) as well as the kinetics of channel activation and deactivation in response to voltage steps (Figure 6*B*).

The full channel model with CDF consists of 3x6 states, as illustrated in Figure 9*B*. The horizontal transitions within the first 2 rows of states are identical to that of the basic unfacilitated model. The last row of states however, represents the facilitated mode, with $\overline{k}_f = k_f \cdot f$, and $\overline{k}_r = k_r / g$, such that the facilitated channel has a larger maximal P_o and increased open times. Additionally, the rate of transitions between the closed states were boosted slightly ($\overline{k}_1 = n \cdot k_1$; $\overline{k}_{-1} = n \cdot k_{-1}$; n = 1.1) to match the kinetics of voltage-activation when channels are fully-facilitated (Figure 6*B*, right). The vertical transitions between states of the 3 rows are determined by the kinetics of CDF (Figure 9). As there was no discernible voltage-dependence to CDF (Figure 8F), the vertical transitions across each of the columns of states are identical, with the exception of the lower-right transition between the open states of rows 2 and 3 where $\overline{k}_3 = k_3 \cdot f$ and $\overline{k}_{-3} = k_{-3} / g$ to maintain the

necessary path-independence. This model is easily represented as a system of differential equations, which was numerically integrated in Matlab with a stiff ODE solver (ode23s). The channel open probability P_o is the probability of residing in any of the 3 open states, and the simulated macroscopic current is then: $I = P_O \cdot N_{chan} \cdot i_{GHK}$, where i_{GHK} is the voltage-dependent unitary current and N_{chan} represents the number of channels.

In Figure 6, Figure ***7*** and Figure ***9***, Ca^{2+} and transmembrane voltage were supplied as inputs into the model, with the channel currents calculated as above, where:

$$i_{GHK} = \left[\underbrace{a_1 \cdot V \frac{Li_i - Li_o \exp(-V/26)}{1-\exp(-V/26)}}_{i_{GHK,Li}} + \underbrace{a_2 \cdot V \frac{Cs_i - Cs_o \exp(-V/26)}{1-\exp(-V/26)}}_{i_{GHK,Cs}} \right] \cdot \underbrace{\frac{1}{1+\exp[(V-71)/30]}}_{b}$$

Two GHK equations were used because in addition to the inward Li^+ current, there was a significant outward current from internal Cs^+ efflux via open Cav2.1 E3A channels. At more depolarized voltages, outward current declines relative to the expected GHK current (Figure 6*B*), likely due to voltage-dependent pore block, as represented here by a voltage-dependent factor *b*.

Physiological model

A spherical cell/synapse was divided into 3 symmetric compartments, representing a thin shell near the membrane, the cytosol, and a reservoir for Ca^{2+} efflux (Figure 12*D*). Cav2.1 channels provide Ca^{2+} influx into the shell compartment, and the resulting Ca^{2+} can then diffuse into the remaining compartments, informed by the following differential equations:

$$\frac{dCa_{shell}}{dt} = \left(Ca_{flux} + k_{cs} Ca_{cyto} - k_{sc} Ca_{shell} \right) / V_{shell}$$

$$\frac{dCa_{cyto}}{dt} = \left(k_{sc} Ca_{shell} + k_e Ca_{rest} - Ca_{cyto} \left[k_{cs} + k_e \right] \right) / V_{cyto}$$

where $Ca_{flux} = -I_{Ca} / 2F$, where F is Faraday's constant, and $I_{Ca} = P_O \cdot N_{chan} \cdot i_{GHK,Ca}$. P_O is determined from the 18 state model described above, where V was provided as an input, and Ca_{shell} used as the Ca^{2+} signal to facilitate Cav2.1 channels. This 20 state model was then solved numerically with Matlab using the stiff ODE solver as before (App. IA. Simulation code for 2 compartment model).

Results

Minimizing Ca^{2+} block of Li^+ currents carried by Cav2.1 channels

Figure 2*A* summarizes our initial attempt to utilize Ca^{2+} photouncaging to investigate CaM regulation of Cav2.1 channels fluxing Li^+. Recombinant channels were heterologously expressed in HEK293 cells. The left subpanel displays a robust Li^+ current evoked by step depolarization to -10 mV. Current decays minimally during the voltage step, as expected for channels coexpressed with β_{2a} auxiliary subunits to minimize VDI [21]. Though Ca^{2+}-loaded DM-nitrophen was present in the internal solution, intracellular Ca^{2+} concentration remained low throughout (~0.1 μM) in the absence of ultraviolet (uv) illumination. The right subpanel summarizes the result during a subsequent voltage pulse wherein Ca^{2+} was uncaged by a brief uv pulse (at vertical violet bar). Prior to uncaging, the initial current waveform (black trace) precisely tracked the control response from the left subpanel (reproduced as gray trace), substantiating minimal rundown. By contrast, upon Ca^{2+} uncaging to nearly 10 μM, the current abruptly decreased to a nadir in less than a few milliseconds, followed by slow recovery over tens of milliseconds. The precipitous decrement of current might have initially appeared perplexing, because it occurred far too quickly to be explained by a slow CDI process

[22]. The subsequent recovery of current could have arisen from CDF developing atop this unspecified inhibitory process. By recalling classic studies of ion permeation through L-type Ca^{2+} channels [40, 41], we soon suspected that the Li^{+} flux through Cav2.1 might be exquisitely susceptible to block by Ca^{2+} binding to the pore (right subpanel, cartoon). The quick decline of current upon Ca^{2+} uncaging would accord with the sub-millisecond equilibration of Ca^{2+} block of monovalent currents fluxing through Ca^{2+} channels.

To test this hypothesis, and to minimize the potential confounding effects on characterizing CaM regulation, we investigated Ca^{2+} block of Cav2.1 channels. The fractional decrement of peak Li^{+} currents (evoked by voltage ramps) was characterized as a function of Ca^{2+} added to the bath (Figure 2*B*). Consistent with prior studies on Cav1.2 channels [42], Cav2.1 exhibited a high affinity for Ca^{2+}, with an IC_{50} of ~0.8 μM. Cognizant that permeation is markedly influenced by four specific glutamate residues in the selectivity filter of L-type channels [43], we reasoned that substituting alanines for the corresponding glutamates in Cav2.1 (Figure 2*D*, top) might also attenuate Ca^{2+} block. Accordingly, E→A substitutions were introduced into each of four homologous domains, and Ca^{2+} block was then quantified. Figure 2*C* summarizes the outcome for E→A substitution in domain III (E3A). Indeed, Ca^{2+} block was markedly reduced compared to wild-type Cav2.1. Nearly 30% of the initial current amplitude was maintained despite 104 μM Ca^{2+} in the bath. As expected, the reversal potential for E3A was also left-shifted relative to wild-type, with Cs^{+} being the main intracellular cation. There was also asymmetry with respect to the effects of alanine substitutions in the various Cav2.1 domains (Figure 2*D*, bottom). Notably, E3A was least sensitive to Ca^{2+} block (IC_{50} of 35

μM), whereas mutations in other domains (E1A, E2A, E4A) yielded smaller IC_{50} values of 7, 8, and 10 μM, respectively. Indeed, this rank order was similar to that in $Ca_V1.2$ [43] and Cav1.3 [36]. That said, we chose the E3A variant for subsequent Ca^{2+}-uncaging experiments, presuming that the confounding decrement of current in Figure 2*A* (right subpanel) was indeed due to Ca^{2+} block.

Steady-state Ca^{2+} sensitivity of $Ca_V2.1$ CaM regulation revealed by Ca^{2+} uncaging

Before performing Ca^{2+} uncaging experiments with the E3A variant, we first verified that the pore mutation did not significantly alter CDF—a plausible expectation given that the molecular determinants of CDF reside within the channel carboxy terminus rather than the permeation pathway [22, 30, 31]. For this purpose, we utilized a well-established prepulse protocol to coarsely estimate CDF as triggered by Ca^{2+} fluxing through channels themselves [22, 44]. Corresponding wild-type $Ca_V2.1$ currents are displayed in Figure 3*A*. Upon presentation of an isolated test pulse to 5 mV (left subpanel), the activation of Ca^{2+} current follows a biphasic timecourse. The first phase comprises an initial rapid increase of current owing to the comparatively speedy voltage activation of channels, initially all in their 'unfacilitated' condition (Figure 3*A*, τ_{fast}). A subsequent second phase of current increase (Figure 3*A*, τ_{slow}) reflects the slower facilitation of channel opening by ongoing Ca^{2+} entering through channels. To characterize facilitation more fully, we examined facilitation produced during brief voltage prepulses, using a variation of this initial protocol. In the example shown (Figure 3*A*, right subpanel, black trace), the test-pulse current following a prepulse exhibits a much reduced slow-activation phase, because many channels have already undergone facilitation during the prepulse. For

reference, the test-pulse current without prepulse has been reproduced in gray. The extent of prepulse facilitation (the metric *CDF*) turns out to be proportional to the area between black and gray traces ΔQ (right subpanel, shaded red area) divided by τ_{slow} (from left subpanel) [44]. Plots of *CDF* as a function of prepulse voltage then yield the bell-shaped relation shown in Figure 3*B*, averaged over multiple E3A (red) and wild-type (black) $Ca_V2.1$ currents. This bell-shape accords with *CDF* reflecting a genuine Ca^{2+}-triggered process, because *CDF* mirrors the amplitude of prepulse Ca^{2+} currents elicited at various prepulse voltages [45]. When Ba^{2+} was substituted as charge carrier, the bell-shaped augmentation was largely eliminated (Figure 3*B*, open symbols), because Ba^{2+} binds poorly to CaM [46]. Importantly, the *CDF* profiles for E3A and wild-type $Ca_V2.1$ channels were indistinguishable, permitting use of the E3A construct to gauge CDF in subsequent Ca^{2+} uncaging experiments.

Turning in earnest towards Ca^{2+} uncaging, we initially measured the effects of Ca^{2+} elevation on a splice variant of $Ca_V2.1$ thought to exhibit far weaker CDF (EFb -47) [44]. Results from this construct explicitly test whether the E3A pore mutation suffices to minimize Ca^{2+} block effects in the precise configuration of uncaging protocols. Figure 3*C* (left subpanel) displays exemplar Li^+ currents through EFb -47 channels bearing the E3A pore mutation. Ca^{2+} remains low throughout (~0.1 μM) in the absence of uncaging. Upon Ca^{2+} uncaging to just over 1 μM (middle subpanel, vertical violet bar), Li^+ current appeared unchanged, demonstrating minimal Ca^{2+} block. Yet stronger Ca^{2+} elevation (right subpanel) also negligibly impacted Li^+ current, substantiating insignificant Ca^{2+} block at levels approaching 10 μM. Likewise, Ca^{2+} elevations over a similar range

produced imperceptible effects on a related EFb +47 splice variant of $Ca_V2.1$ [44] (Figure 4). Having confirmed that Ca^{2+} block effects were adequately blunted by E3A mutation, we focused again on the prototypic $Ca_V2.1$ variant (EFa -47) that exhibits robust CDF, as shown earlier in Figure 3*A*-*B*. Figure 3*D* summarizes the effects of Ca^{2+} uncaging, specifically on a version of these prototypic channels containing the E3A mutation. The left subpanel shows steady Li^+ current evoked without Ca^{2+} uncaging. By contrast, upon Ca^{2+} uncaging (middle subpanel, gray-shaded region), a nearly two-fold augmentation of current was rapidly produced (red-shaded region). Stronger and more sustained Ca^{2+} elevation (right subpanel) induced a still larger and more sustained elevation of current. These effects directly substantiate a fully legitimate CDF process, one that appears considerably larger than previously estimated by indirect means such as in Figure 3*B* [22, 23, 47]. Data obtained from multiple cells and uncaging experiments are summarized in Figure 3*E*. For EFa -47 channels, the steady-state relation between *CDF* and $[Ca^{2+}]$ defines a Hill function with a maximum of 2.22, midpoint EC_{50} of approximately 0.5 μM, and Hill coefficient n_{Hill} of 1.8 (red curve). *CDF* was here defined as the ratio of currents evoked before and after Ca^{2+} uncaging at steady state. For EFb -47 channels, *CDF* was negligibly elevated for $[Ca^{2+}]$ up to 10 μM (gray relation). Reassuringly, CDF was essentially eliminated in EFa -47 channels coexpressed with a dominant-negative CaM molecule (Figure 3*F*), confirming CaM as the sensor for CDF as found in prior studies [22, 35, 47]. Moreover, no appreciable CDF was observed for EFa -47 channels bearing an isoleucine to alanine substitution within an IQ domain in the $Ca_V2.1$ carboxy terminus

(Figure 2*F*), an outcome that fits with previous characterization of this mutant using more indirect measures [22].

Beyond the large magnitude of CDF, another unexpected observation was the absence of CDI in all the records shown thus far [22, 48]. In particular, there was no hint of current decline in both EFb -47 and EFa -47 records after Ca^{2+} uncaging to several micromolar (Figure 3C and **Figure *3***D, rightmost subpanels). It turns out that Ca^{2+} elevations greater than 10 μM are required to trigger CDI (Figure 5*E*). Owing to technical constraints, it was difficult to uncage Ca^{2+} to such large concentrations, while reliably achieving low baseline [Ca^{2+}] prior to uncaging. For this reason, we focused the remainder of experiments on the CDF process, which could be studied in isolation because of the low Ca^{2+} sensitivity of the CDI mechanism.

Contrasts between fully facilitated and non-facilitated channels at various voltages

Thus far, our data nicely revealed features of CDF, but only as viewed during voltage steps to -10 mV. We therefore sought next to examine the contrasts in opening at different voltages, as exhibited by unfacilitated $Ca_V2.1$ channels compared to those fully facilitated by Ca^{2+}. In particular, Li^+ currents evoked before Ca^{2+} uncaging (Figure 6*A*, left subpanel, black trace) reflect the opening and closing of channels governed by transitions among closed states (*C*) and a predominant open configuration (*O*) that comprise an unfacilitated mode of gating (middle subpanel, top) [35]. Most of the transitions are voltage dependent, while the final transition into *O* is largely voltage insensitive [35]. By contrast, after strong uncaging to fully facilitate channels with respect to Ca^{2+}, Li^+ currents evoked by an identical voltage pulse protocol (right

subpanel, black trace) would largely reflect transitions within a 'facilitated' gating mode (middle subpanel, bottom), where opening is favored over that in the unfacilitated mode (middle subpanel, top). That said, a critical feature of CDF concerns the relative advantage in opening between these two gating modes over a range of voltages.

Figure 6*B* displays the results of a protocol that specifically reveals this advantage. Tail Li^+ currents through $Ca_V2.1$ EFa E3A -47 channels were evoked immediately following a series of brief activating voltage pulses ranging from -30 to 50 mV, initially in the absence of Ca^{2+} uncaging (blue dots). The response for the activating pulse to -10 mV (highlighted in gray zone) is displayed on an expanded scale in Figure 6*A* (left subpanel, black trace). Plotting the peak amplitude of these tail currents as a function of voltage in corresponding activating pulses (Figure 6*C*, blue symbols and relation) yields the steady-state voltage activation profile of unfacilitated channels. For convenient comparison across cells, the data have been normalized to the largest tail current amplitude. Immediately following acquisition of this initial set of tail currents, current was evoked by a sustained voltage pulse to -10 mV , and Ca^{2+} uncaged to 3.4 μM to produce the usual facilitation of current (note events in Figure 6*B* near vertical violet bar) as illustrated previously in Figure 3*D*. As this level of Ca^{2+} saturated facilitation with respect to Ca^{2+} (Figure 3*E*), subsequent acquisition of tail currents (Figure 6*B*, red dots) sufficed to approximate the steady-state voltage activation profile for fully facilitated channels (Figure 6*C*, red symbols and relation). These data have been normalized to the maximal tail current before Ca^{2+} uncaging. Viewed in this way, one can readily deduce that CDF produces both an absolute upward scaling of the voltage activation profile

(Figure 6*C*, effect *a*) and a hyperpolarizing shift of the relation (effect *b*) [35]. Dividing the relations for facilitated and unfacilitated channels yields the fractional increase in open probability produced by CDF as a function of voltage (Figure 6*D*). These results were fully corroborated over multiple cells (Figure 6*E* and **Figure *6F***). Intriguingly, the net result of effects *a* and *b* is to produce a Boltzmann-like *CDF* profile (Figure *6D* and **Figure *6F***). To confirm that all effects under this protocol were strictly related to CDF, we replicated experiments using Cav2.1 EFb E3A -47 channels (which lack facilitation, Figure 3*C*), and observed a complete absence of CDF behavior (Figure 6*G* and **Figure *6H***).

Two notable features merit attention. First, the magnitudes of *CDF* at voltages less than zero are remarkably larger than expected from traditional protocols (Figure 3*A*), reaching values exceeding two. *CDF* values of this order are consistent with the results of Ca^{2+} uncaging experiments performed earlier during step depolarizations to -10 mV (Figure 3*D*, right subpanel), but now we can appreciate that such large facilitation would extend over a range of voltages less than zero. Second, physiological activation of Cav2.1 channels, such as during neuronal action potentials, would draw heavily from this hyperpolarized voltage range, so one might expect the effects of CDF to be especially pronounced during physiological spike activation.

To explore further this inference about action-potential behavior, we constructed channel kinetic mechanisms corresponding to the gating modes in Figure 6*A*, using the parameters in Figure 7*A*. Quantitative simulations reproduced several important features: the whole-cell current waveforms evoked in Figure *6A* (red curves) and **Figure *6B***; all

the voltage activation and CDF profiles in Figure 6*E*-**Figure *6F***; and published single-channel open times [35]. As previously suggested [35], differences between unfacilitated and facilitated gating modes were mainly restricted to the rightmost opening and closing steps according to fixed scaling factors *f* and *g* (Figure 6*A* and Figure 7*A*). With our gating mechanisms thus established, we turned to predictions of the maximal impact of CDF on currents evoked by physiological action-potential waveforms, taken from cerebellar Purkinje neurons [47]. For channels resident within the unfacilitated mode, simulated activation by a typical neuronal action potential (Figure ***7B***, left, red curve) predicts well the actual $Ca_V2.1$ current (black trace). Upon maximal Ca^{2+}-induced conversion towards the facilitated mode, the corresponding simulated current increases by nearly twofold (Figure ***7B***, right, red curve). This outcome fits with the intuition that CDF could strongly enhance currents during physiological activation. That said, Figure 7*C* tests experimentally for CDF during trains of action-potential waveforms. For an exemplar cell, with Li^+ currents fluxing through recombinant $Ca_V2.1$ EFa E3A -47 channels, evoked currents remained steady over multiple action-potential waveforms in the absence of Ca^{2+} uncaging (left subpanel), substantiating stability of the preparation. By contrast, upon uncaging to a saturating level of Ca^{2+}, corresponding currents increased by nearly twofold (Figure ***7C***, right subpanel). The impact of CDF on current is further emphasized by the expanded timebase plots of exemplar currents evoked just before and after Ca^{2+} uncaging (Figure ***7B***, black traces, corresponding to Figure ***7C*** responses shaded in green). As well, data averaged over many cells confirmed that full induction of CDF can strongly amplify spike current (Figure ***7D-E***).

Voltage insensitivity of Ca^{2+}-driven conversion to facilitated channels

Still uncertain, though, is whether strong interconversion towards the facilitated mode would actually be produced by physiological Ca^{2+} elevations. In particular, only at a fixed potential of -10 mV has the Ca^{2+} dependence of CDF been determined (Figure 3*E*). This Ca^{2+} sensitivity reflects the steady-state distribution of channels between upper and lower modes (Figure 8*A*), as governed by vertical transitions aligned with closed and/or open states most heavily populated at -10 mV (say, for example, k_d transitions). At different potentials, the most frequented closed and/or open states may differ considerably, emphasizing different vertical transitions (for example, k_a, k_b, k_c, k_e or k_f) that may yield alternative Ca^{2+} sensitivities, knowledge of which would be fundamental to appreciating the physiological relevance of CDF. Accordingly, we took advantage of the fact that it takes 1-3 minutes for Ca^{2+} to return to baseline after large Ca^{2+} uncaging events, owing to the relatively slow diffusion of fresh DM-nitrophen from the pipet into the cell interior. Figure 8*B* displays results for an exemplar cell undergoing such a protocol. As Ca^{2+} gradually recedes, we could continue periodic assessment (every 5-30 seconds) of voltage activation curves using a tail-current protocol (format as in Figure 6*B*). In this manner, we could assess the status of near steady-state channel partitioning between unfacilitated and facilitated configurations, over a broad range of smoothly waning Ca^{2+} levels. Most of this partitioning would occur at the holding potential of -80 mV (5-30 second sojourns), punctuated by significant visitations to a wide range of other potentials during the tail-current protocols that last ~0.5 seconds. If the vertical transitions in Figure 8*A* were appreciably different, we would anticipate *CDF* –Ca^{2+} relations that deviate considerably from those obtained at -10 mV throughout (Figure

3*E*). Figure 8*C* displays the result for this exemplar cell, where fold increase in tail-current amplitude is plotted versus Ca^{2+} concentration. For example, for responses following brief activation to 0 mV, tail-current amplitudes denoted by the cyan symbols in Figure 8*B* were normalized to that seen before Ca^{2+} uncaging, and these normalized amplitudes were then plotted versus corresponding Ca^{2+} concentrations to yield the cyan *CDF* –Ca^{2+} relation in Figure 8*C*. Analogous analysis yielded *CDF* –Ca^{2+} curves relating to different activation potentials (other colored relations in Figure 8*C*). Normalizing all of these relations yields the single Ca^{2+} response profile in Figure 8*E*, which is indistinguishable from the steady-state function obtained at -10 mV in Figure 3*E* (EC_{50} = 0.5 μM and n_{Hill} = 1.8). Note that currents obtained after Ca^{2+} returned to baseline (Figure 8*B*, far right) were indistinguishable from those obtained before uncaging (Figure 8*B*, far left), excluding appreciable run-down phenomena and permitting fold changes in tail current amplitude to be directly attributed to CDF. Averaged data from seven cells fully corroborated this finding (Figure *8D* and **Figure *8F***). Thus, Ca^{2+}-driven conversions between unfacilitated and facilitated conformations (Figure 8*A*) are insensitive to membrane potential.

Dynamic Ca^{2+} responsiveness of CDF

The key remaining biophysical unknown concerned the temporal response of CDF to Ca^{2+} signals. Uncaging Ca^{2+} during a sustained voltage step allows direct millisecond resolution of CDF induction. In this regard, a first important set of observations emerged when we overlaid multiple current traces of Cav2.1 EFa E3A -47 channels (Figure 9*A*, gray waveforms) corresponding to different-sized Ca^{2+} steps exceeding 3 μM (Figure

9*A*). The black waveform shows the average of these. These data indicate that CDF can evolve quickly with a monoexponential time constant τ ~2 ms (Figure 9*A*), and that τ does not further decrease with larger Ca^{2+} steps. The latter property implicates the existence of a Ca^{2+}-independent and rate-limiting step leading to the facilitated mode of gating. Thus, a minimal mechanism would in reality require three modes of gating (Figure 9*B*), as follows. The transition from unfacilitated (top) to prefacilitated modes of gating (middle) would be Ca^{2+} dependent. Notably, the propensity for opening in these two gating modes would be largely equivalent, so as to allow for the rate limitation of a subsequent transition (Ca^{2+} independent) to an outright facilitated mode with enhanced opening (bottom). These three gating modes correspond to previously described conformations underlying CaM regulation of Ca^{2+} channels [49] (see legend, Figure 9*B*). Elucidating this simple three mode mechanism would permit prediction of how CDF evolves in response to the complex Ca^{2+} waveforms present physiologically in neurons.

Towards this end, Ca^{2+} photo-uncaging was used to deliver Ca^{2+} waveforms with variable magnitudes and time courses as shown in Figure 9*C* (top), while simultaneously recording currents (below). The different Ca^{2+} waveforms were produced by varying uv pulse intensity. Weak pulses liberated comparatively few Ca^{2+} ions that could be rapidly buffered via free cytoplasmic DM-nitrophen, yielding weaker Ca^{2+} transients with rapid decays. Strong pulses would uncage so many Ca^{2+} ions that free cytoplasmic DM-nitrophen would be consumed, such that minutes might be required to rebind released Ca^{2+} ions via fresh DM-nitrophen diffusion from the pipette. Moderate uv pulses would produce Ca^{2+} transients with intermediate properties.

Given this rich data set, we proceeded to constrain parameters by fitting Ca^{2+} waveforms with double exponential functions (Figure 9*C*, top, smooth curves), and then inputting these functions to the kinetic scheme in Figure 9*B*. The parameters (Figure 9*D*) were then adjusted until simulated outputs matched both experimental kinetic data (Figure 9*C*, bottom, smooth curves) and steady-state Ca^{2+} responsiveness (Figure 9*E*, smooth curve) determined earlier. By repeating this procedure for additional cells, we obtained the final mean parameters for intermodal exchange shown in Figure 9*D* (from n = 8 cells). Our model also incorporates a Ca^{2+} concentration exponent n_{Hill} of 1.8. Rate constants within the various gating modes were identical to those established in Figure 7*A*. With these parameters, the simulated timecourse of CDF in response to Ca^{2+} step increases and decreases can be shown in Figure *9F* and **Figure *9G***. The maximal predicted 'on' and 'off' responses exhibits respective time constants of 2 ms and 31 ms, seemingly fast enough for physiological relevance.

Devising strategies for resolving native $Ca_V2.1$ properties

Use of the $Ca_V2.1$ E3A construct permitted Li^+ currents to be directly studied in Ca^{2+} uncaging experiments, allowing unprecedented access to the Ca^{2+}-dependence of CDF. However, the strategies employed thus far could not be used to study native $Ca_V2.1$ channels, owing to the problem of Ca^{2+} block (i.e., Figure 2*A*, right subpanel). It was nonetheless crucial to substantiate our biophysically rigorous understanding of recombinant $Ca_V2.1$ channels in the native setting.

We thus devised a 'sparse-tail-current' variant of our Ca^{2+} uncaging protocol. If Ca^{2+} (rather than Li^+) could be used as the charge carrier, then little Ca^{2+} block would occur

upon uncaging of intracellular Ca^{2+} [42], even in native channels with wild-type permeation properties. However, Ca^{2+} influxing through channels would constitute local Ca^{2+} sources with attendant elevations of nanodomain Ca^{2+} that would be difficult to measure with bulk-distributed Ca^{2+} fluorescent dyes [34]. Indeed, this complexity, as outlined earlier in Figure 1*A*, motivated our use of Li^{+} currents in the first place. If, however, we applied only brief voltage pulses separated by comparatively long periods of quiescence (Figure 10*A*, left subpanel), we might be able to assess the propensity for opening (during brief voltage pulses) while the spatially uniform global Ca^{2+} component specifies the extent of CDF during brief opening (owing to the predominance of global Ca^{2+} during long interpulse segments). Importantly, bulk-distributed Ca^{2+} fluorescent dye would reliably measure this global Ca^{2+} concentration (Figure 10*A*, right subpanel). Figure 10*B* schematizes further the hoped-for outcome. Brief (1-ms) activating pulses would be alternatively presented to 0 and 60 mV, with long (>50 ms) quiescent periods separating pulses. Also obtained would be corresponding tail currents, proportional to the open probability at the end of 1-ms activating voltage pulses. These tail currents (cyan dot after 0-mV pulse; rose dot after 60-mV pulse) would thus monitor the extent of CDF at two key points along the overall voltage-dependent profile of CDF (i.e., Figure 6*F*). A critical point is that although Ca^{2+} influx during channel openings would sharply increase nanodomain Ca^{2+} via a local component (Figure 10*B*, yellow shaded area), this local perturbation would rapidly dissipate such that nanodomain Ca^{2+} during most of the period preceding voltage pulses would be equivalent to the global Ca^{2+} component measured by bulk Ca^{2+} fluorescent dye (gray shaded area). Moreover, it is precisely this preceding

global Ca^{2+} concentration that should specify the extent of CDF reflected in tail currents, because the change in CDF that occurs during a 1-ms voltage pulse should be small, as follows. First, the Ca^{2+} entry during 1-ms activating voltage pulses would be limited, because there would be little Ca^{2+} influx during 60-mV steps (near the reversal potential), or voltage activation would be comparatively slow at 0 mV. Second, even under saturating Ca^{2+}, CDF can increase no faster than specified by a ~2 ms time constant (Figure 9*A*), and the voltage pulses in question only spans 1 ms. All told, if all these scenarios were to hold true, then the *CDF* versus Ca^{2+} relations deduced via this variant protocol should accord with those obtained more directly using Li^{+} currents fluxing through Cav2.1 E3A channels (Figure 8*E*).

Figure 10*C* displays the experimental outcome for an exemplar cell expressing Cav2.1 EFa -47 channels fluxing Ca^{2+}. The voltage pulse protocol is displayed in the top row, with the global Ca^{2+} concentration readout shown below in the second row. The same resulting tail-current traces are displayed at two gains in the remaining rows. In the third row, the higher-gain trace shows well the smaller tail currents following pulses to 0 mV (black segments of trace, cyan dots), and the segments containing the larger tail currents after 60-mV pulses are plotted in gray and clipped to the frame. The fourth row plots the lower-gain trace optimized for display of larger tail currents after 60-mV pulses (black segments of trace, rose dots), and the segments containing tails after 0-mV pulses are shown in gray. With this setup in mind, the manifestation of CDF can be readily appreciated. Prior to Ca^{2+} uncaging, tail currents remain steady between pulses, and immediately following Ca^{2+} uncaging (vertical violet bar) there is strong CDF as

demonstrated by the substantial enhancement of tail currents following pulses to both 0 and 60 mV. This observation is corroborated by expanded time base displays of current immediately before and after the uncaging event (Figure 10*D*), which support nearly twofold enhancement of tails following 0-mV pulses. Following Ca^{2+} uncaging, global Ca^{2+} levels gradually relaxed over the course of minutes, as shown in the remainder of Figure 10*C* via display of four time segments punctuated by time-axis breaks. Likewise, tail current amplitudes diminished in parallel with the decline of Ca^{2+}, where the horizontal cyan and rose lines reference amplitudes before Ca^{2+} elevation. Importantly, upon return of Ca^{2+} to baseline (rightmost segment), tail currents were closely similar to those prior to uncaging, allowing interpretation of intervening currents in terms of CDF apart from run down. We thus plotted fold increase in tail current amplitude as a function of instantaneous Ca^{2+} to obtain the quasi-steady-state *CDF*–Ca^{2+} correlations in Figure 10*E*. Reassuringly, the functions for 0 and 60 mV closely resembled the actual steady-state relations obtained using the original protocol with Li^{+} currents (Figure 8*C*). As expected, the larger CDF observed for 0- versus 60-mV pulses reflects the voltage dependence of unfacilitated versus facilitated modes of gating (Figure 6*A* and **Figure *6F***). Indeed, amplitude scaling of both plots (Figure 10*F*) furnished a normalized relation tightly resembling that obtained at steady state (Figure 8*E*), with nearly the same half Ca^{2+} level (EC_{50} = 0.3 μM) and slightly diminished Hill coefficient of 1.5 (versus EC_{50} = 0.5 μM and n_{Hill} = 1.8 in Figure 8E). The small differences here may reflect imperfect exclusion of the local Ca^{2+} component in this sparse-tail-current protocol. Similar analysis in multiple cells furnished equivalent results (Figure 10*G* and **Figure *10H***).

Thus, we established that this sparse-tail-current protocol could assess *CDF*–Ca^{2+} relations with near quantitative precision while employing Ca^{2+} as the charge carrier through channels.

Prominent CDF in native $Ca_V2.1$ in cerebellar Purkinje neurons

Thus armed, we turned to characterization of CDF in native $Ca_V2.1$ currents in cerebellar Purkinje neurons, renowned for the predominance of these Ca^{2+} channels [50]. Acutely dissociated neurons from P14-18 mice were used (Figure 11*A*, inset), so as to maximize the presence of EFa versus EFb variants (~85% EFa transcripts [51]). Although the presence of CDF in these currents had been detected previously [25, 47], the means of quantification and characterization were indirect, the extents of CDF seemingly modest, and the quantitative dependence on Ca^{2+} concentration unspecified.

To corroborate directly the existence of strong CDF in native $Ca_V2.1$ channels, we evoked native Ca^{2+} currents with trains of action-potential waveforms (50 Hz), and simultaneously introduced Ca^{2+} uncaging events. During baseline conditions without uncaging (Figure 11*A*), currents appeared steady (horizontal cyan line) from the beginning to end of a train, and Ca^{2+} readouts remained constant near 0.1 μM. Thus, it appeared that the local Ca^{2+} component (Figure 10*B*) was insufficient to trigger appreciable CDF under these conditions. In the next train of action potentials (Figure 11*B*), currents were initially indistinguishable from those in the previous train (horizontal cyan line). However, upon strong uncaging to several micromolar Ca^{2+} (vertical violet bar), pronounced ~1.6× amplification of currents was produced, indicating the potential for large native CDF matching that observed in prior figures for recombinant channels.

To quantify the Ca^{2+} sensitivity of this CDF, we applied the sparse-tail-current protocol devised in Figure 10*B*. Results from a single exemplar Purkinje neuron are reported in Figure 11*C-E*, using the same display format as in Figure 10*C*. Absent Ca^{2+} uncaging (Figure 11*C*), currents remained steady and bulk Ca^{2+} persisted near 0.1 μM, all indicative of a stable baseline. Upon moderate Ca^{2+} uncaging (Figure 11*D*), currents exhibited modest amplification which decayed as Ca^{2+} levels relaxed. Upon intense Ca^{2+} uncaging (Figure 11*E*), currents manifested marked and sustained enhancement of nearly twofold for -10-mV pulses. This modulation can be further appreciated in the expanded time base plots of currents immediately before and after uncaging (Figure 11*F*). Because of slow drift of currents following large Ca^{2+} elevations in these neurons, we could not in general compile traces over the course of minutes (as in Figure 10*C*), but were instead limited to analyzing data from several trace segments, each containing a different-sized uncaging event. Using this strategy in this exemplar neuron, plots of fold enhancement of tail currents as a function of Ca^{2+} yields the *CDF*–Ca^{2+} relations in Figure 11*G*. Reassuringly, these bear a strong resemblance to those obtained for recombinant Cav2.1 channels (Figure 10*E*), and amplitude scaling yields an aggregate normalized relationship (Figure 11*H*, EC_{50} = 0.5 μM and n_{Hill} = 1.3) closely similar to that of recombinant channels undergoing the same protocol (Figure 10*F*). Population data from n = 5 neurons replicate these findings (Figure 11*I* and **Figure *11F***), and firmly establish strong native CDF, whose parameters largely accord with those of recombinant channels. Compared to recombinant channels, the slightly diminished maximal *CDF* in native currents (Figure

11*I*) fits with a modest fraction of Cav2.1 EFb variants that lack CDF (~15% transcripts [51]).

Discussion

This study reveals the long-sought speed and extent of Ca^{2+}-dependent facilitation (CDF) of Cav2.1 channels. Using Ca^{2+} photouncaging as a signal generator of intracellular Ca^{2+}, we demonstrate that CDF is considerably larger, faster, and more Ca^{2+} sensitive than previously imagined. This performance profile postures Cav2.1 channels for extensive CDF under neurophysiological activity, offering the potential for strong activity-dependent adjustment to downstream signaling pathways. Additionally, the new and extensive Ca^{2+} and electrophysiological constraints furnished by our approach significantly advance mechanistic understanding of this regulatory system, which is formalized by a realistic biophysical model of Cav2.1 gating and CDF. Overall, these findings hold key implications for CDF in relation to structural determinants, functional mechanisms, and neurophysiological roles, as follows.

Structural determinants of CDF

During baseline characterization via Ca^{2+} uncaging, important aspects of the structural basis of CDF were refined. From qualitative measures of CDF, the carboxy tail of Cav2.1 has previously been suggested to be essential for CDF [31], just as for CaM regulation of Cav1 channels [52]. In this regard, a first advance in the present study was to revise prior suggestions that alternative splicing at exon 37 (encoding EFa or EFb modules within the carboxy tail) enables or disables CDF, but in a manner modulated by inclusion of exon 47 on the extreme channel carboxy terminus [44]. Using the higher-resolution Ca^{2+} uncaging

assay used here, we now demonstrate that the EFa/EFb module fully switches CDF on and off (Figure 3*C*, *E*), and that inclusion or exclusion of exon 47 does not affect Cav2.1 CDF (Figure 4). This simplifying feature strengthens the view that CDF may be exquisitely tuned in the human brain, since the EFa:EFb variant ratio is regulated in an age-specific, region-specific and sex-specific manner [44, 47, 51]. Furthermore, the complete switching of CDF at the EF locus offers a simple strategy to create transgenic animal models that lack or possess CDF, thereby aiding clarification of its physiological role. Similarly, one wonders whether diseases of aberrant splicing [53] might act at this EF locus. Second, beyond the EF element, substituting alanine for the central isoleucine within the IQ domain also eliminates CDF under the Ca^{2+} uncaging regime (Figure 3*G*), affirming further the critical role of the IQ element for CDF [22, 30, 31]. Third, the elimination of CDF by Ca^{2+}-insensitive mutant CaM_{1234} (Figure 3*F*) consolidates the role of Ca^{2+}-free CaM (apoCaM) as a resident CDF sensor bound to the channel carboxy terminus [22]. Finally, our finding of a Hill coefficient near 2 for *CDF*–[Ca^{2+}] relations (Figure 3*E* and Figure 8*F*) affirms prior conclusions that CDF is predominantly triggered by Ca^{2+} binding to the C-terminal lobe of CaM [22], which contains two highly cooperative EF-hand Ca^{2+} binding sites.

Advances in functional mechanisms

Our experimental paradigm confirms an ‘enhanced-opening’ mechanism of CDF whereby the maximal open probability of channels at saturating depolarization is enhanced [35]. That said, with the improved resolution in the present study, we now also detect a 5-mV left shift of the voltage activation relation (Figure *6C*, *E*) [26], whereas our

previous study lacked the sensitivity to detect this effect [35]. Though modest, it is this shift in voltage dependence that amplifies CDF significantly during physiological patterns of activation (Figure *6D* and *F*, Figure *7B* and *E*). These effects could be almost entirely explained by enhancing the last transition to the open state (Figure 6, k_f) while reducing the closing rate from the open state (k_r), leaving all other transitions largely unchanged.

As for the Ca^{2+}-induced transitions among unfacilitated and prefacilitated modes of gating (vertical arrows in Figure 9*B*), we find that these transitions can be approximated as the same, regardless of position along the activation pathway. Physically speaking, this feature suggests that Ca^{2+} binding to calmodulin is largely insensitive to the status of voltage activation. We further suggest that the transitions between prefacilitated and facilitated gating modes are also insensitive to voltage activation, except when the actual open states are reached (Figure 9*B*, also see Methods | Simulations – $Ca_V2.1$ channel models). This latter attribute hints that the binding of Ca^{2+}/CaM to a channel effector site is mainly sensitive to the final concerted opening step, rather than movements of voltage-sensing paddles.

Together, all these features can be incorporated within an explicit biophysical gating mechanism (Figure 9*B*), arguably the most extensive for calmodulin regulation of any channel. Indeed, the CDI of Cav1.2/3 channels, as triggered by the C-lobe of CaM [54, 55], may turn out to be described by an analogous mechanism (Figure 9*B*), except that k_f / k_r in the bottom gating mode would be reduced, and the modes would be described as non-inactivated, pre-inactivated, and inactivated.

A final novel functional feature concerns the stark contrast in the Ca^{2+} sensitivity of CDF versus CDI (Figure *5E*), far more than imagined previously. Whereas the EC_{50} for CDF is ~0.5 μM Ca^{2+}, the corresponding value for CDI would approximate ~30 μM. Recalling the prior finding that Cav2.1 CDI is triggered by Ca^{2+} binding to the N-lobe of CaM [22], as well as the mechanistic framework developed previously [49], we suggest that this relative Ca^{2+} insensitivity of CDI may arise from a far greater channel affinity for the N-terminal lobe of apoCaM versus Ca^{2+}/CaM. The wide divergence of EC_{50} values for CDF and CDI would allow facilitation to develop in isolation over sustained periods of Ca^{2+} elevation less than ~10 μM [56]. The induction of CDI might be reserved for clustered channels positioned at local subcompartments where Ca^{2+} may achieve concentrations exceeding that in bulk cytoplasm. The prior observation that CDI is only apparent in a fraction of cerebellar Purkinje neurons [47] may coincide with this large EC_{50} for CDI, and the sporadic attainment of these Ca^{2+} levels. Estimated Ca^{2+} for CDF and CDI in the calyx of Held (respective EC_{50} values of 2.5 μM and 6 μM [57]) differ somewhat from our estimates, but these experiments used young P8-P10 rats hosting a mixture of Cav2.1, Cav2.2, and Cav2.3 channels [58-60].

Neurophysiological Implications

Our experiments in cerebellar Purkinje neurons (Figure 11) not only substantiate the existence of CDF within these neurons [47], but reveal just how large CDF can be in this setting (~1.8x in Figure 11*G* and *I*). The similarity of Ca^{2+} sensitivity for CDF in these neurons (EC_{50} ~0.5 μM in Figure 11*J*), compared to that for recombinant channels (~0.5 μM in Figure 8*F*), further supports CaM as the Ca^{2+} sensor for CDF [47]. Given the

prominence of Ca^{2+} signaling in Purkinje neurons [61], along with the primacy of Cav2.1 as Ca^{2+} entry portal in this locus [7, 62], CDF of Cav2.1 channels seems particularly well poised for physiological influence in this context.

A first potential role concerns recent exciting findings that coordinated motor control may require cerebellar Purkinje neurons to spontaneously fire simple spikes with regular and uniform rate [63-65]. Empirically speaking, this requirement can be verified by correlating altered coordination with pharmacological and/or genetic modulation of spike regularity [64]. From the theoretical standpoint, the need for constant spiking may relate to promising synchronization encoding mechanisms wherein motor information is conveyed through selected deep cerebellar nuclei only at appropriate times, in particular when inhibitory inputs from multiple convergent Purkinje neuron become synchronized [66, 67]. Achieving such synchrony, for example following a shared complex-spike phase reset among convergent Purkinje neurons, would require these neurons to manifest a regular and uniform spontaneous simple spike rate [68]. We can connect this fundamental substrate of motor coordination to Cav2.1 CDF by noting that phasic activation of small-conductance Ca^{2+}-activated K channels (SK channels), which drives after-hyperpolarization following spikes, is a prime determinant of spontaneous simple spike rate [69]. Moreover, the dominant trigger of SK channels involves preferential Ca^{2+} signaling from colocalized Cav2.1 channels fluxing peak Ca^{2+} current during the repolarizing phase of spikes (e.g., Figure 7*B*) [69, 70]. Accordingly, attenuation of either SK or Cav2.1 channels, such as in heritable forms of ataxia, begets rate variability and motor dyscoordination [63]. All that said, we suggest that Cav2.1 CDF would represent

an elegant servo-control mechanism to enhance regularity of simple spike rate, as follows. Perturbations yielding transient rate increases would heighten global Ca^{2+}, boost $Ca_V2.1$ CDF and SK channel activation, and thereby downwardly restore rate towards an intrinsic setpoint (Figure 12*A*). Conversely, transient rate decreases would diminish global Ca^{2+}, dampen CDF and SK channel activity, and consequently upwardly restore rate (Figure 12*B*). Thus, $Ca_V2.1$ CDF may play a role in optimizing motor coordination. Indeed, the postnatal developmental increase of CDF (owing to switching from $Ca_V2.1$ EFb to EFa variants [47]) closely parallels the timecourse of motor coordinative improvements [71].

Secondly, given the predominance of $Ca_V2.1$ channels in triggering transmitter release across the CNS [3, 9, 72], CDF might contribute presynaptically to short-term synaptic plasticity, either to boost outright facilitation or to buffer the extent of depression. Indeed, trains of spike-like depolarizations induce large 1.8× facilitation of rat (P9-P17) $Ca_V2.1$ currents in the calyx of Held [73]. Given that neurotransmitter release probability is proportional to Ca^{2+} concentration to the third or fourth power [29, 74], this CDF could orchestrate order-of-magnitude (Figure 11*G-J*, 1.8^4 ~10) modulation of synaptic strength. Of note, reports of weaker CDF in calyceal $Ca_V2.1$ currents restricted experiments to conditions minimizing vesicle depletion [75], with the result that Ca^{2+} levels (252 nM) may have been insufficient to recruit much of CDF by our Ca^{2+} sensitivity measures (arrow in Figure 12*C*). More broadly, we anticipate a strong contribution of CDF to synaptic plasticity at other synapses. When we embedded our *in silico* channel mechanism (Figure 9*B*) within a generic three-compartment spherical

terminal (Figure 12*D*), we reproduced (Figure 12*E*) both the kinetics and magnitude of experimentally observed CDF in the calyx [73]. In this scheme (Figure 12*D*), Ca^{2+} enters initially through Cav2.1 channels into a submembranous shell compartment, followed by rapid diffusion into a larger cytosolic compartment. This cytosolic space is in turn connected with an efflux pathway into a large axonal compartment set at resting Ca^{2+}. When simulated $Ca_V2.1$ channels are opened with a 1-ms depolarization to -20 mV (Figure 12*E*, top), peak Ca^{2+} currents (bottom row) caused Ca^{2+} elevation in the shell and cytoplasmic compartments (Ca_{shell} and Ca_{cyto} in rows 3-4). The shell compartment experiences large Ca^{2+} spikes coincident with channel opening, each followed by rapid relaxation to the cytosolic (global) pedestal. The global cytoplasmic Ca^{2+} rises with each stimulus during a 100-Hz train, eventually reaching a quasi plateau where Ca^{2+} influx matches its efflux. Interestingly, even though CaM physically senses local Ca^{2+} within the shell, it is the global Ca^{2+} component in this compartment that drives much of CDF when brief spikes are sufficiently spread out. Thus, the timecourse of CDF tracks the rise and fall of global Ca^{2+}, much as in our sparse-tail protocol (Figure 10*A*). Altering stimulus frequency reveals that CDF quickly decays with slower pacing (Figure 12*F*), projecting substantial contributions to short-term synaptic plasticity.

In this light, it is worth quantifying how CDF fits with other facilitatory mechanisms underlying rapid synaptic upregulation (Figure 12*C*). The red curve plots the Ca^{2+} sensitivity of outright neurotransmitter release, presumably as specified by Ca^{2+} binding properties at C2 sites on synaptotagmin [74]. High Ca^{2+} concentrations (>10 μM) may be required for this process, such as produced by clusters of Cav2.1 channels near

neurotransmitter vesicle complexes [76]. However, accumulation of 'residual' presynaptic Ca^{2+} from prior activity [10], insufficient to trigger release, may nonetheless preload unspecified high-affinity Ca^{2+} binding sites [77, 78] to boost ensuing release according to a residual Ca^{2+} (RC) mechanism (Figure 12*C*, dashed *RC* curve). Here, our Ca^{2+} uncaging experiments project Cav2.1 CDF as another relevant system, in this case with intermediate Ca^{2+} sensitivity (Figure 12*C*, black curve). Of note, synapses in the calyx of Held of transgenic mice lacking Cav2.1 (compensated by Cav2.2) exhibit little to no short-term synaptic facilitation [79]. That said, we do not believe that RC and CDF need be one and the same process, as synaptic paired-pulse facilitation in the calyx of Held can be observed in the absence of CDF [75]. Overall, it seems likely that CDF and RC processes will differ in their relative contributions at various synapses, and the RC category may encompass several as-yet-loosely-defined processes. That said, our extensive biophysical clarification of Cav2.1 CDF, in both recombinant and native currents, now leaves little doubt that this Ca^{2+} regulation will figure prominently in the CNS.

Figures

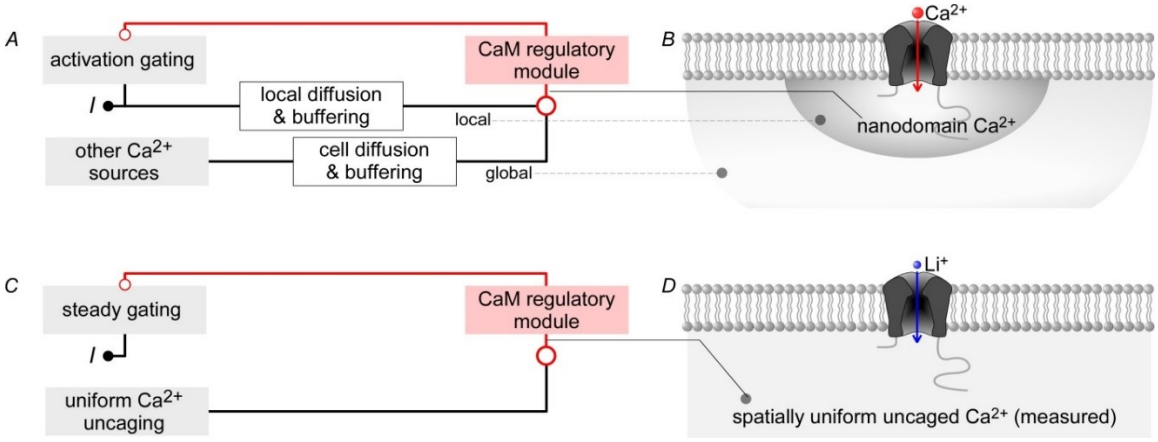

Figure 1

Figure 1: **Ca^{2+} sources influencing $Ca_V2.1$ opening.**

A-B, Ca^{2+}/CaM regulation of channels arises from a superposition of local and global Ca^{2+} sources. $Ca_V2.1$ channels stimulated by a voltage step gives rise to Ca^{2+} current *I*, which is proportional to channel activation gating. The Ca^{2+} current, shaped by local diffusion and buffering, creates a local nanodomain Ca^{2+} signal that feeds into the CaM regulatory module, which bestows feedback modulation on activation gating. Concurrently, Ca^{2+} from distant channels shaped by cell diffusion and buffering, creates a global Ca^{2+} signal which is also an input into the CaM regulatory module. *C-D*, simplified scheme employing Li^+ as charge-carrier. Stimulating $Ca_V2.1$ here gives rise to Li^+ current *I*, which is allowed to reach steady-state (steady gating) before Ca^{2+} is uncaged to influence the CaM regulatory module. Importantly, in this system, Ca^{2+}-uncaging generates the only Ca^{2+} signal, which is spatially uniform and measurable.

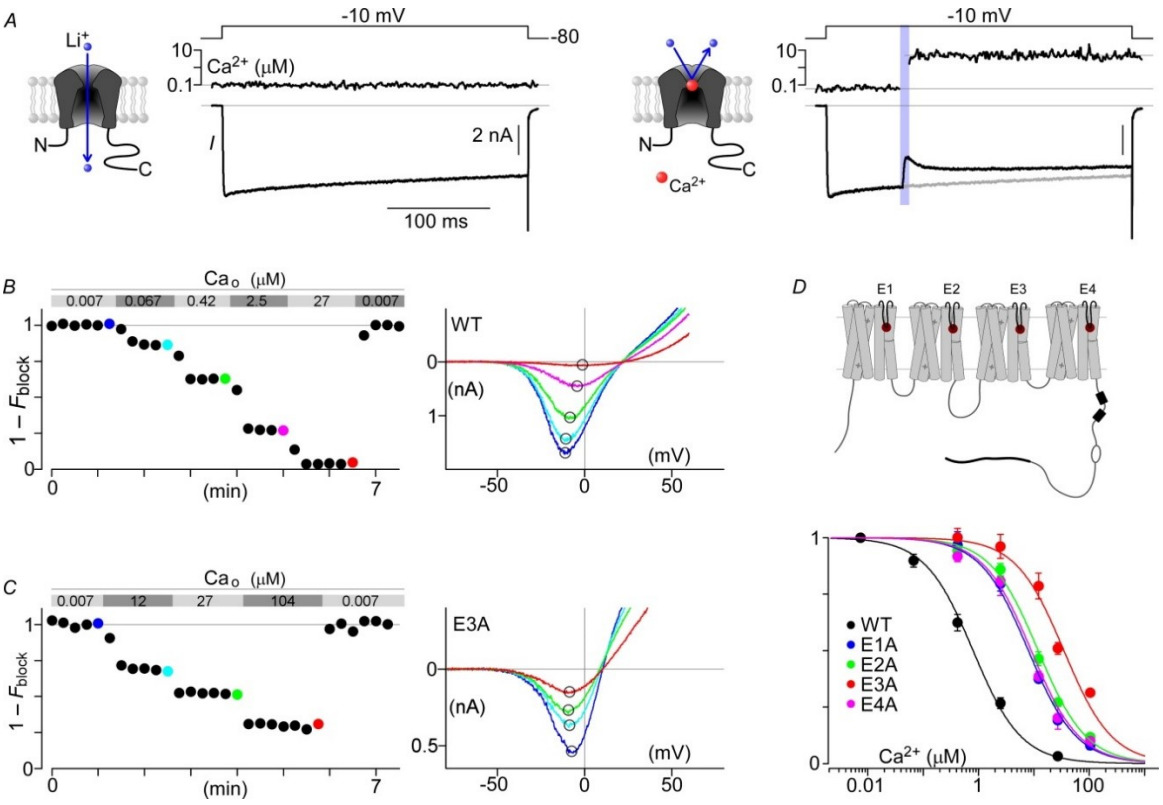

Figure 2

Figure 2. Relieving Ca^{2+} block of Li^{+} current.

A, Li^{+} permeation through $Ca_V2.1$ is blocked by Ca^{2+}. $Ca_V2.1$ EFa fluxes Li^{+} current in response to a step depolarization at baseline intracellular Ca^{2+} (*left*). A brief uv pulse (violet bar) induces a step-like increase in intracellular Ca^{2+} in the midst of the voltage step, resulting in a rapid block of Li^{+} current, followed by CDF (*right*). *B*, characterization of Ca^{2+} block. Li^{+} currents from voltage ramps were obtained in the presence of varying amounts of Ca^{2+} in the bath (Ca_o). Normalized peak currents show the decrement in peak currents of $Ca_V2.1$ in response to Ca^{2+}. Exemplar *I-V* traces (*right*) from the last trace of each Ca^{2+} solution (colored dots, *left*) show explicitly the reduction in current amplitude without shifting of the reversal potential. *C*, similar traces for $Ca_V2.1$ E3A show a reduced Ca^{2+} sensitivity to block (*left*), and a left-shifting of the reversal potential (*right*). *D*, cartoon of $Ca_V2.1$ channel highlighting the four glutamates (E1-E4) forming the selectivity filter (*top*). Population (mean±SEM) Ca^{2+}-block data for $Ca_V2.1$ (*bottom*): WT (IC_{50} = 0.8 μM; n = 7), E1A (IC_{50} = 7 μM; n = 5), E2A (IC_{50} = 8 μM; n = 5), E3A (IC_{50} = 35 μM; n = 5), E4A (IC_{50} = 10 μM; n = 5). Fits to binding curves all have Hill coefficient of 1.

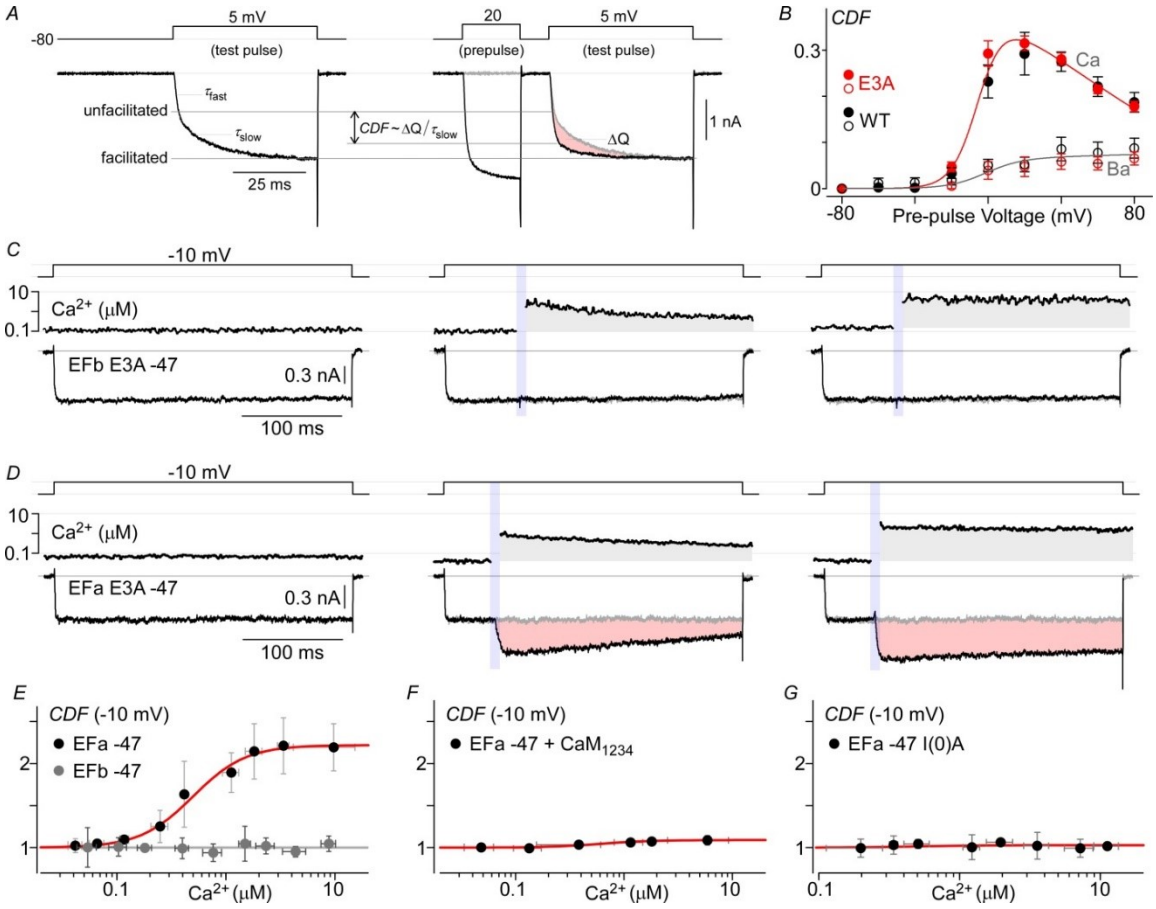

Figure 3

Figure 3. Steady-state Ca^{2+} sensitivity of $Ca_V2.1$ CaM regulation revealed by Ca^{2+} uncaging.

A, characterization of E3A's CDF using the prepulse protocol. With Ca^{2+} as the charge carrier, a step depolarization to 5 mV induces an inward current with rapid and slow components due to a superposition of voltage-activation and CDF (*left*). With a preceding voltage step to 20 mV, sufficient Ca^{2+} enters to partially facilitate the channels, such that the subsequent step to 5 mV has a markedly blunted slow component (*right*). The area between the two current traces (ΔQ), divided by τ_{slow}, gives an approximation of the *CDF* triggered by the prepulse. *B*, population data (mean±SEM) for *CDF vs* prepulse voltage of the pore mutant E3A (red, $n = 9$) and WT (black, $n = 8$). The open circles denote the use of Ba^{2+} as the charge-carrier (incapable of triggering CDF), while the filled circles denote the use of Ca^{2+}. *C*, E3A mutant permeating Li^{+} current shows negligible Ca^{2+} block. Ca^{2+} uncaging during a voltage step of a $Ca_V2.1$ E3A splice-variant (EFb, -47) induces negligible block, and no CDF (*middle*, *right*). *D*, Ca^{2+} uncaging with the EFa variant (-47) showcases large and rapid induction of CDF (*middle*, *right*). *E*, population data for $Ca_V2.1$ EFa -47 (black dots, mean±STD, $n = 12$ cells), quantifying *CDF* as a function of different uncaged Ca^{2+} concentrations during voltage steps to -10 mV yields a

Ca^{2+} sensitivity curve, with EC_{50} = 0.6 μM and n_{Hill} = 1.8. Population data for $Ca_V2.1$ EFb -47 (gray dots, mean±STD, n = 17 cells) show complete absence of CDF. *F*, similar experiments with $Ca_V2.1$ EFa -47 coexpressed with CaM_{1234}, a mutant CaM incapable of binding Ca^{2+}, reveal elimination of CDF (mean±STD, n = 7 cells). *G*, similar experiments with $Ca_V2.1$ EFa -47 carrying I(0)A mutation at the IQ domain show a knockout of CDF (mean±STD, n = 8 cells).

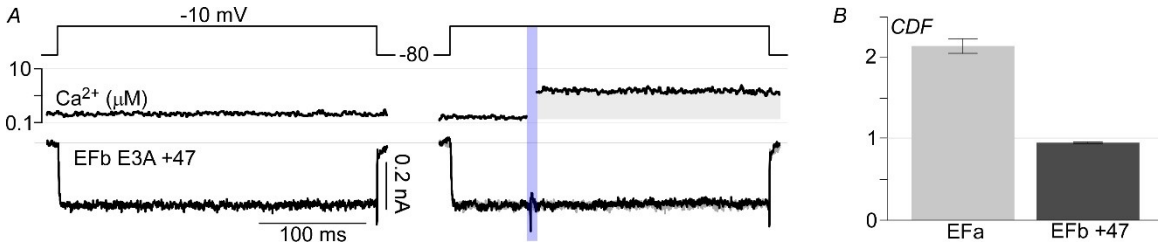

Figure 4

Figure 4. Cav2.1 EFb +47 lacks CDF.
A, Li^+ currents through Cav2.1 EFb +47 are stable at baseline Ca^{2+} (*left*). Upon Ca^{2+} uncaging, currents are negligibly affected (*right*). *B*, population data from $n = 5$ cells where Ca^{2+} was uncaged >1 μM, showing disruption of *CDF* (*right*). Data for Cav2.1 EFa (*left*) is from Figure 3*E*, averaged for $Ca^{2+} > 1$ μM.

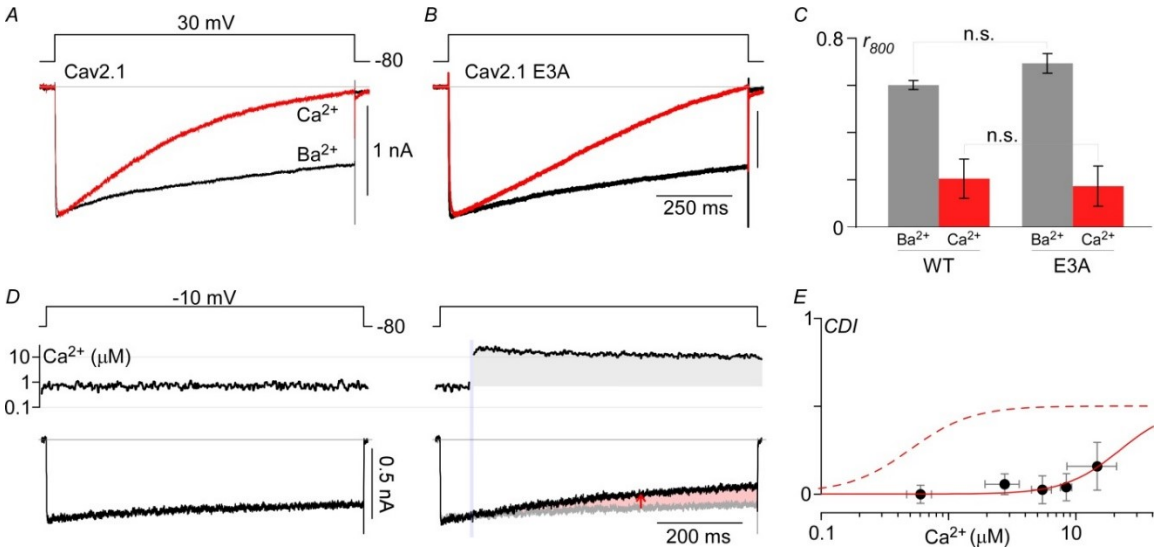

Figure 5

Figure 5. High Ca^{2+} requirements for CDI.

A, exemplar currents from Cav2.1 permeating either Ca^{2+} (red) or Ba^{2+} (black) during a 1 sec depolarization to 30 mV. *B*, similar protocol for Cav2.1 E3A. In both cases, 40 Ca^{2+}/40 Ba^{2+} (mM) were used to increase current density. *C*, population data for WT and E3A channels, plotting the ratio of currents at 800 ms to peak current for Ba^{2+} and Ca^{2+} (mean±sem). Both WT and E3A have $n = 5$ cells each, with comparison done using unpaired Student's *t* test ($p > 0.05$). *D*, Li^{+} currents through Cav2.1 EFb E3A channels are stable at ~0.7 μM (*left*). Uncaging Ca^{2+} to ~14 μM elicits a small amount of CDI (*right*). *E*, compiling traces from n = 19 cells where Ca^{2+} was uncaged to various levels yields a Ca^{2+} dose-response curve for *CDI* (red solid line), defined here as $CDI = r_{400,before\ uncaging} - r_{400,after\ uncaging}$ (red arrow in *D*), where $r_{400} = I(t = t_{uvpulse}) / I(t = t_{uvpulse+400ms})$. Data points shown are mean±std, where fit is a Hill function with $n_{Hill} = 2$, $IC_{50} = 22$ μM. Fit for CDF's Ca^{2+} sensitivity is reproduced as red dotted line for comparison.

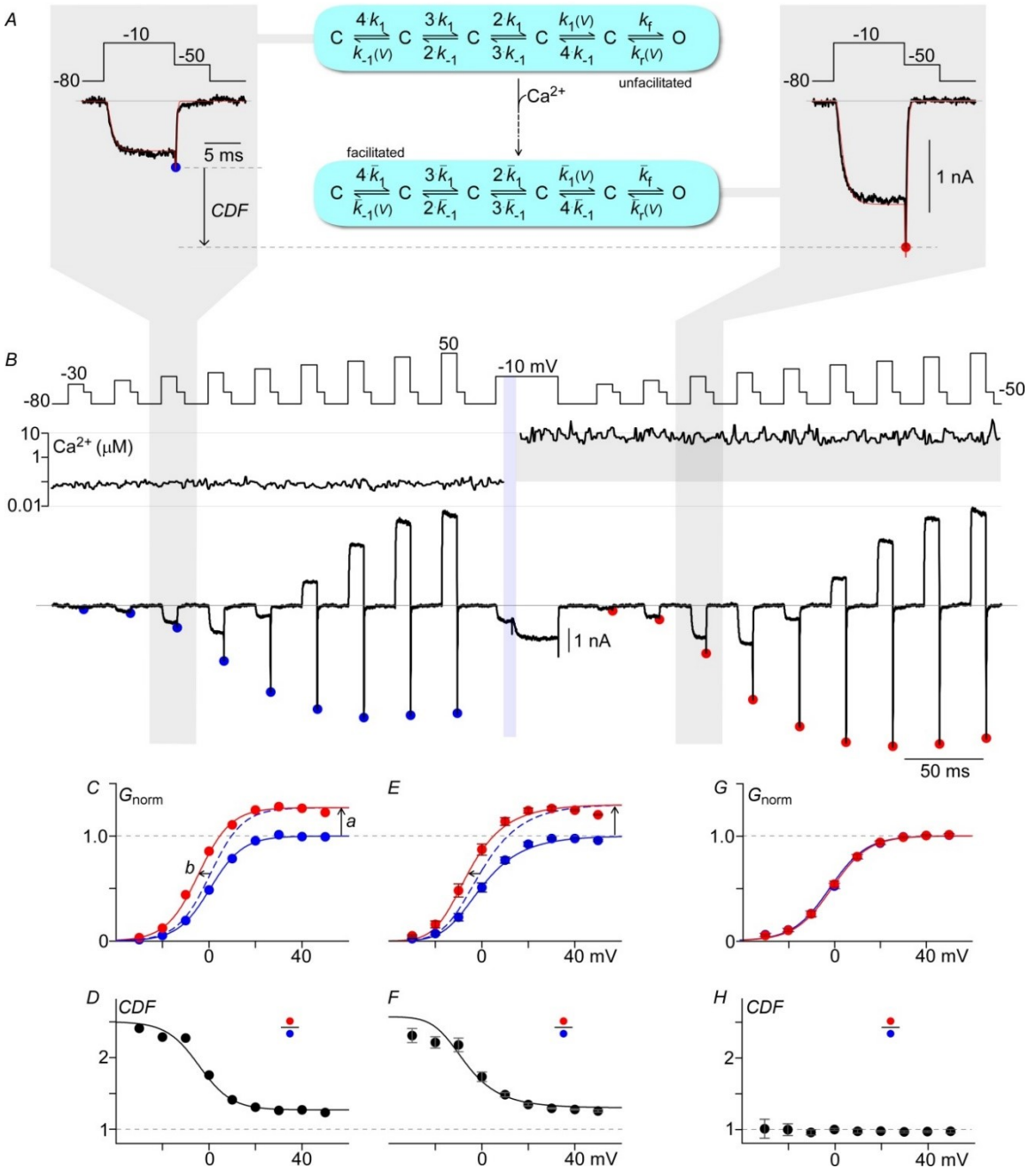

Figure 6

Figure 6. Contrasts between facilitated and non-facilitated channels at various potentials.

A-B, protocol to define *G-V* curves of unfacilitated and facilitated channels. Li^+ tail currents from Cav2.1 EFa E3A are elicited at -50 mV after 10 ms step depolarizations ranging from -30 to 50 mV. The peak tail currents before (blue) and after (red) Ca^{2+}

uncaging are marked with filled circles. Ca^{2+} is uncaged during a 40 ms voltage step at -10 mV to monitor progression of CDF. *A*, expanded view of tail currents (black) after voltage steps to -10 mV before (*left*) and after (*right*) Ca^{2+} uncaging. A kinetic model of Cav2.1 channel gating is diagrammed (*middle*), with simulation results (red, *left* and *right*) overlaid on top of the data. Parameters for model shown in Figure 7. *C*, exemplar *G-V* from *B*. The peak of the tail currents, normalized to the peak currents after steps to 40 and 50 mV at baseline Ca^{2+} (blue), provides an assessment of how the voltage activation curve changes in response to Ca^{2+} (red). Two effects are seen: a frank increase in G_{max} (*a*), and a left-shift of ~5 mV (*b*). *D*, *CDF vs* voltage for exemplar. *E-F*, similarly, population data (mean±SEM) with $n = 7$ cells where Ca^{2+} was raised >3 μM. Fit lines are from simulations of the channel model diagrammed in Figure 6A (*middle*). *G-H*, population data for Cav2.1 EFb E3A (mean±SEM) with Ca^{2+} uncaged to >3 μM for $n = 7$ cells.

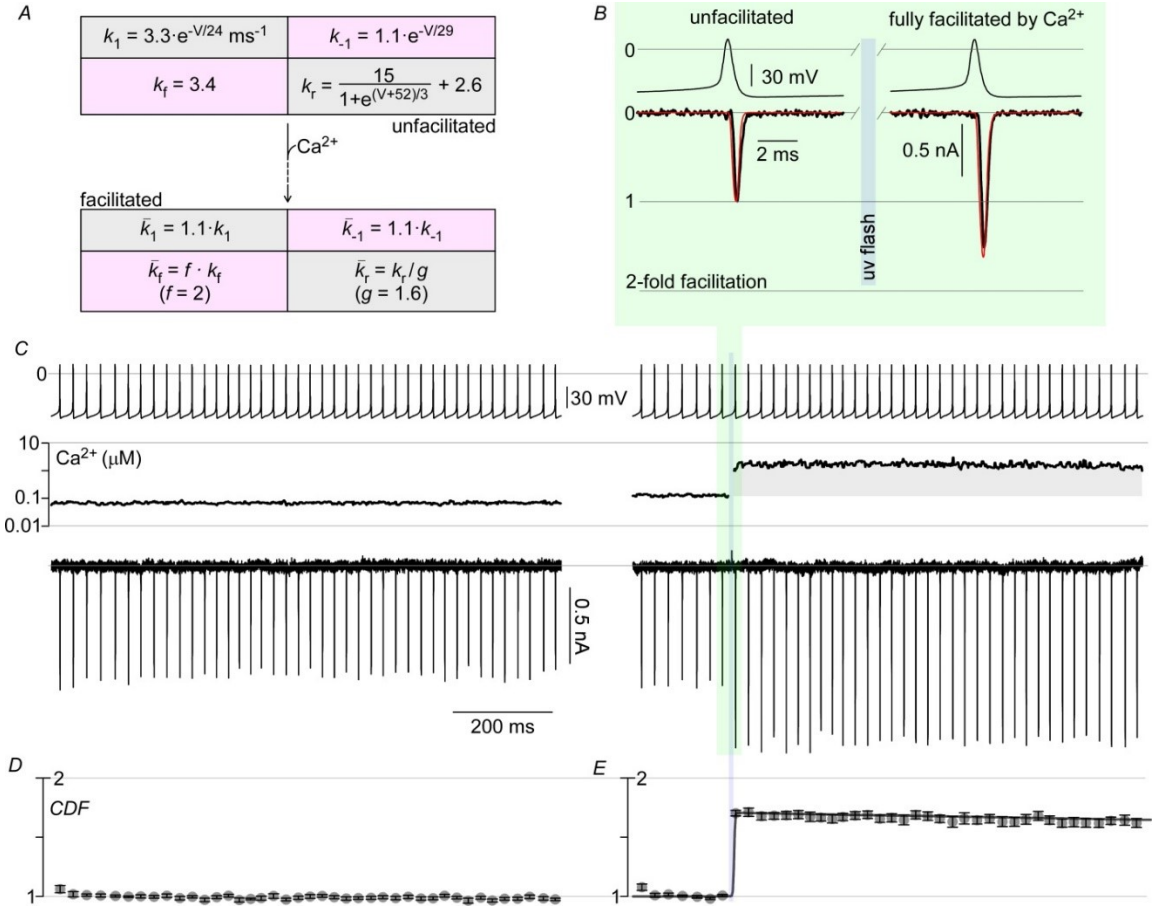

Figure 7

Figure 7. Physiologically-relevant action potential waveforms have significant CDF.

A, parameters of channel model diagrammed in Figure 6A in the unfacilitated and facilitated states. *B*-*C*, $Ca_V2.1$ E3A permeating Li^+ currents are stimulated with action potential waveforms recorded from cerebellar Purkinje neurons (courtesy of Raman, I.). Stimulated currents are steady at baseline Ca^{2+} (*C*, *left*). Ca^{2+} uncaging in the midst of the ~50Hz train causes large enhancement of Li^+ current (*C*, *right*). An expanded view of traces before and after Ca^{2+} elevation (black) are shown in *B*, with simulation results overlaid (red). *D*-*E*, population data (mean±SEM) for peak Li^+ current as a function of time with n = 5 cells.

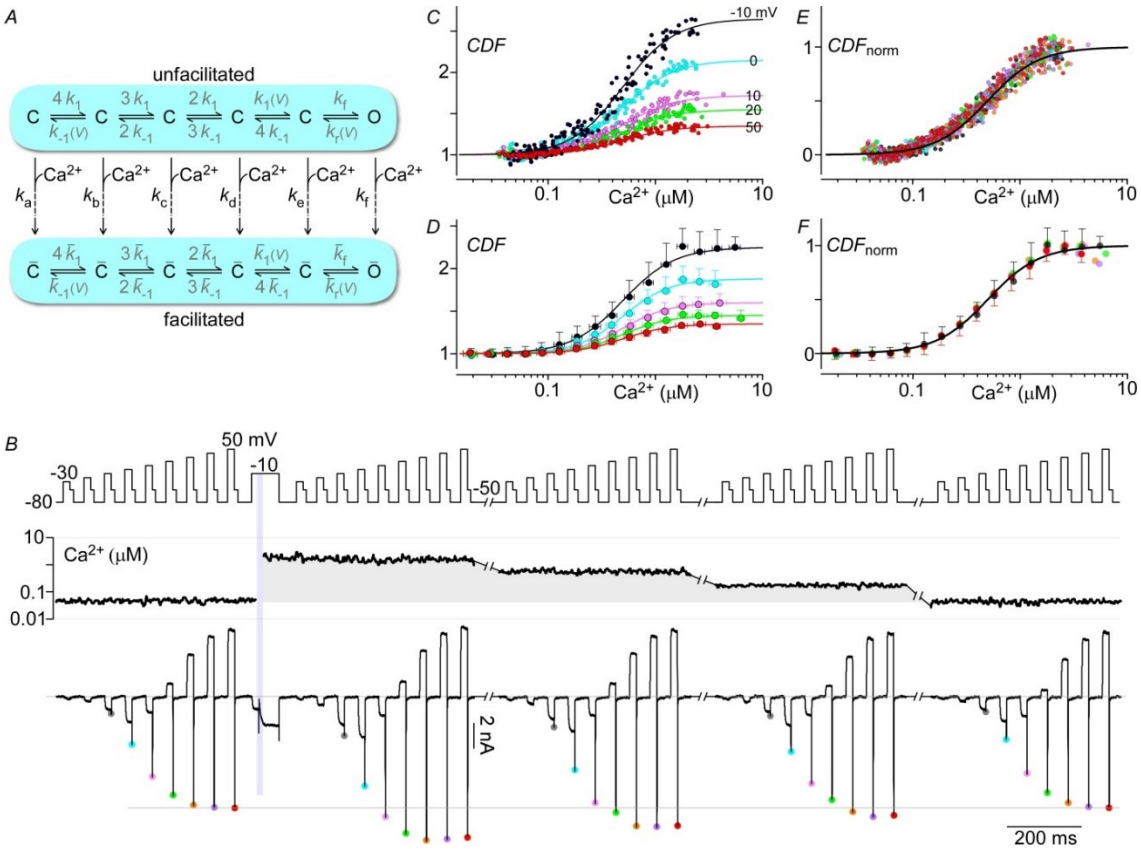

Figure 8

Figure 8. Voltage-insensitivity of Ca^{2+}-driven conversion to facilitated channels.

A, diagram of channel model illustrating the possibility of having multiple rate constants (k_a-k_f) driving channels towards facilitated states. *B*, multiple *G-V* curves are obtained with tail protocols before and after uncaging Ca^{2+}. As the intracellular Ca^{2+} concentration drops over time, the peak tail currents correspondingly decrease. *C*, the peak Li^+ currents plotted against the intracellular Ca^{2+} charts out the steady-state *CDF vs*. Ca^{2+} curves for the voltages assessed. *D*, normalized *CDF* vs Ca^{2+} for voltages from -10 to 50 mV show that CDF at different voltages have the same Ca^{2+} sensitivity. *E-F*, As above, population data (mean±STD) shown for n = 7 cells. All fits are Hill equations with n_{Hill} = 1.8, and EC_{50} = 0.5 μM.

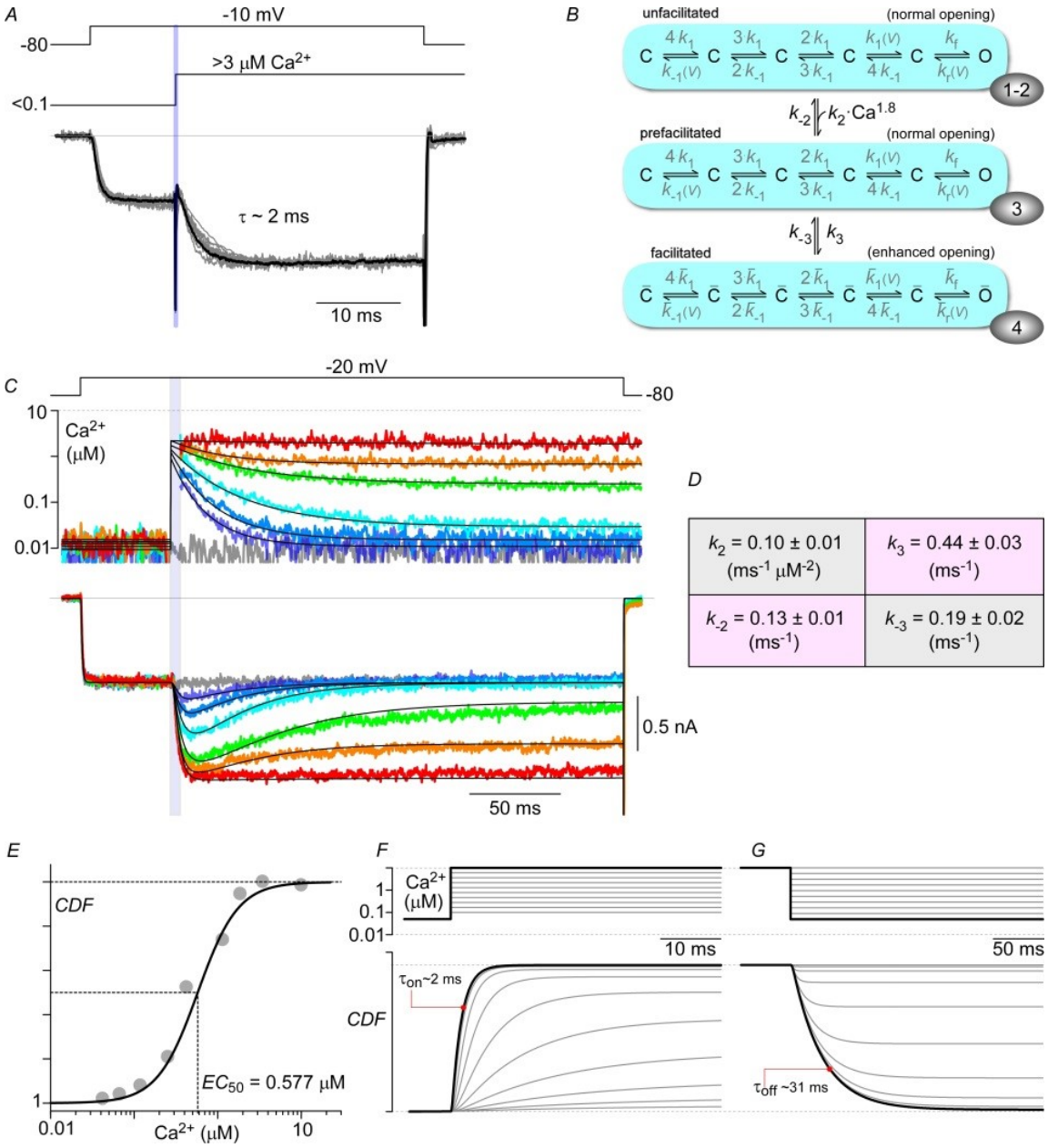

Figure 9

Figure 9. Dynamic Ca^{2+} responsiveness of CDF.

A, saturating on-kinetics of CDF. Individual normalized current records (gray) from 10 cells, where intracellular Ca^{2+} was uncaged to >3 μM, are shown overlaid together with their average (black), fit by a single exponential with $\tau = 2$ ms. *B*, Cav2.1 channel model, where the vertical Ca^{2+}-dependent transitions have 3 states (rows), with transitions between 2nd-3rd rows having no Ca^{2+}-dependence. The correspondence of these three gating modes to previously described conformations underlying CaM regulation of Ca^{2+}

channels [49] is denoted by the numbers within ovoids at the lower right edge of each gating mode. Conformation 1 corresponds to a channel in which the C-terminal lobe of Ca^{2+}-free CaM (apoCaM) is bound to a channel preassociation site; conformation 2 depicts a channel wherein this lobe of apoCaM transiently unbinds from the preassociation site; conformation 3 diplays a channel where this transiently unbound lobe of CaM complexes with Ca^{2+}, and conformation 4 represents the channel after the Ca^{2+}-saturated lobe of CaM has interacted with a channel effector site to produce facilitation. For simplicity, conformations 1 and 2 have been combined in simulations used in the present study. *C*, Exemplar traces from a single cell, where a family of time-varying Ca^{2+} inputs by Ca^{2+} uncaging (second row) trigger a corresponding family of kinetically-distinct CDF records (last row). The smooth black lines on the Ca^{2+} and current axes represent the fit to the Ca^{2+}-waveforms used as the input and the output of the channel model respectively. Parameters for the model are shown to the right (mean±SEM), obtained from fitting $n = 8$ cells. *E*, these parameters yield a Ca^{2+} sensitivity curve matching data from Figure 3E (gray dots). *F*-*G*, simulations of step increases and decreases of intracellular Ca^{2+} show the maximal 'on' and 'off' responses have time constants of 2 ms and 31 ms.

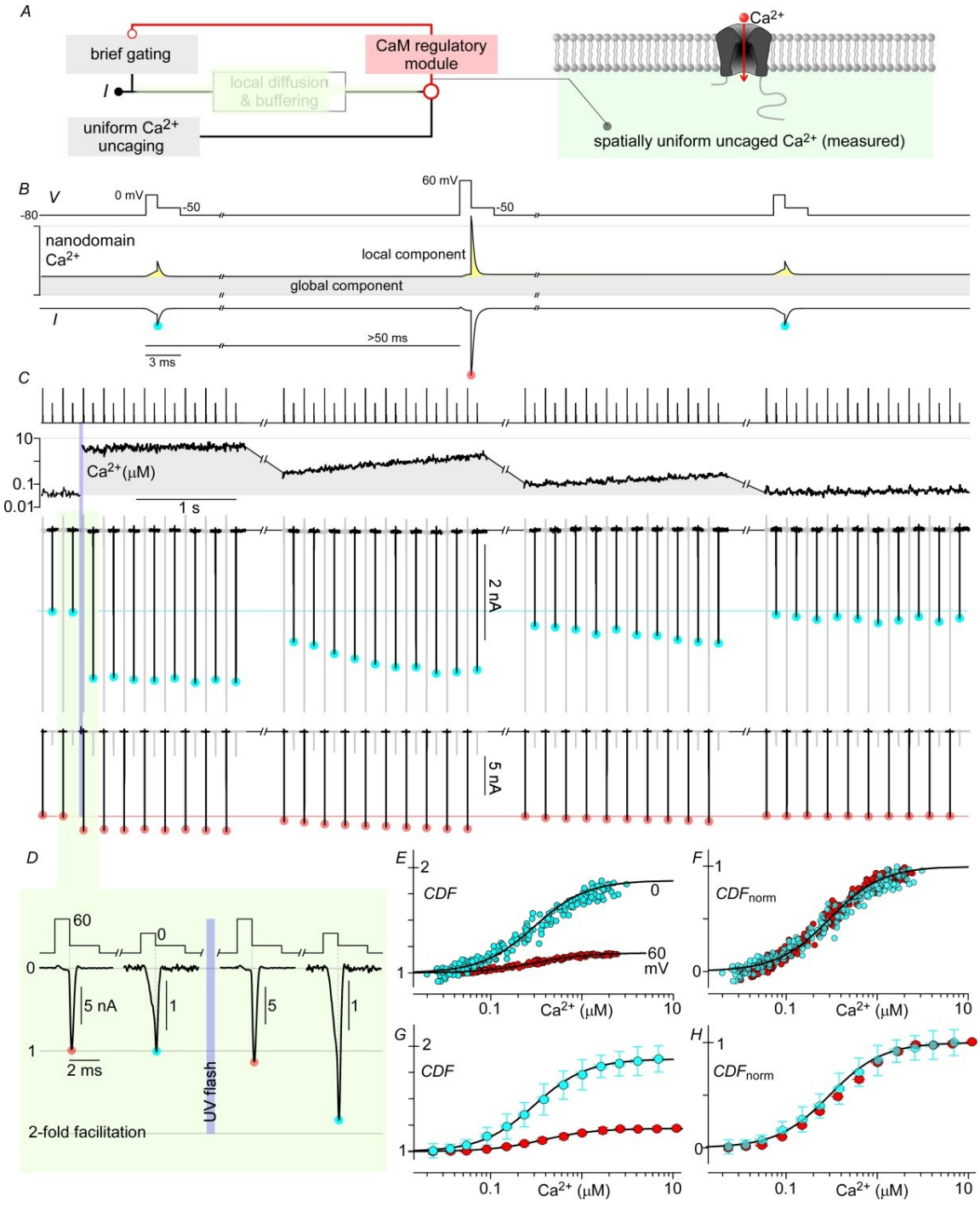

Figure 10

Figure 10. Towards resolving native $Ca_V2.1$ CDF.

A-B, schema outlining strategy to resolve Ca^{2+}-dependence of native Cav2.1 channels. A modified tail current protocol with very brief steps to 0 or 60 mV for 1 ms (brief gating)

before eliciting tail currents at -50 mV minimizes the impact of local Ca^{2+} on the CaM regulatory module whilst still allowing assessment of the channel's state at two voltages. Uniform Ca^{2+} uncaging then provides an independent and measurable input to the regulatory module, pushing channels into the facilitated mode. *C*, Exemplar traces from a single cell, illustrating Ca^{2+} tail currents evolving with time and global Ca^{2+}. Voltage pulse protocol is displayed in the top row, and global Ca^{2+} in the second row. Tail currents triggered by preceding step to 0 mV are scaled up for clarity and shown in black in the third row (tails triggered by preceding 60 mV steps in gray), with cyan dots marking the peaks. The last row shows the tail currents triggered by preceding 60 mV steps in black, with peaks marked by red dots. *D*, expanded time frame views of tail currents before and after Ca^{2+} uncaging, normalized to peak levels at baseline Ca^{2+}. *E*, steady-state *CDF vs* Ca^{2+} curves at the two voltages assessed are obtained by plotting normalized peak tail currents against instantaneous Ca^{2+} levels. *F*, *CDF vs* Ca^{2+} curves of the 2 voltages are normalized, showing the same Ca^{2+} sensitivity. *G-H*, population data (mean±STD) show similar results, with $n = 9$ cells. Fit curves have $n_{Hill} = 1.5$, $EC_{50} = 0.3$ μM.

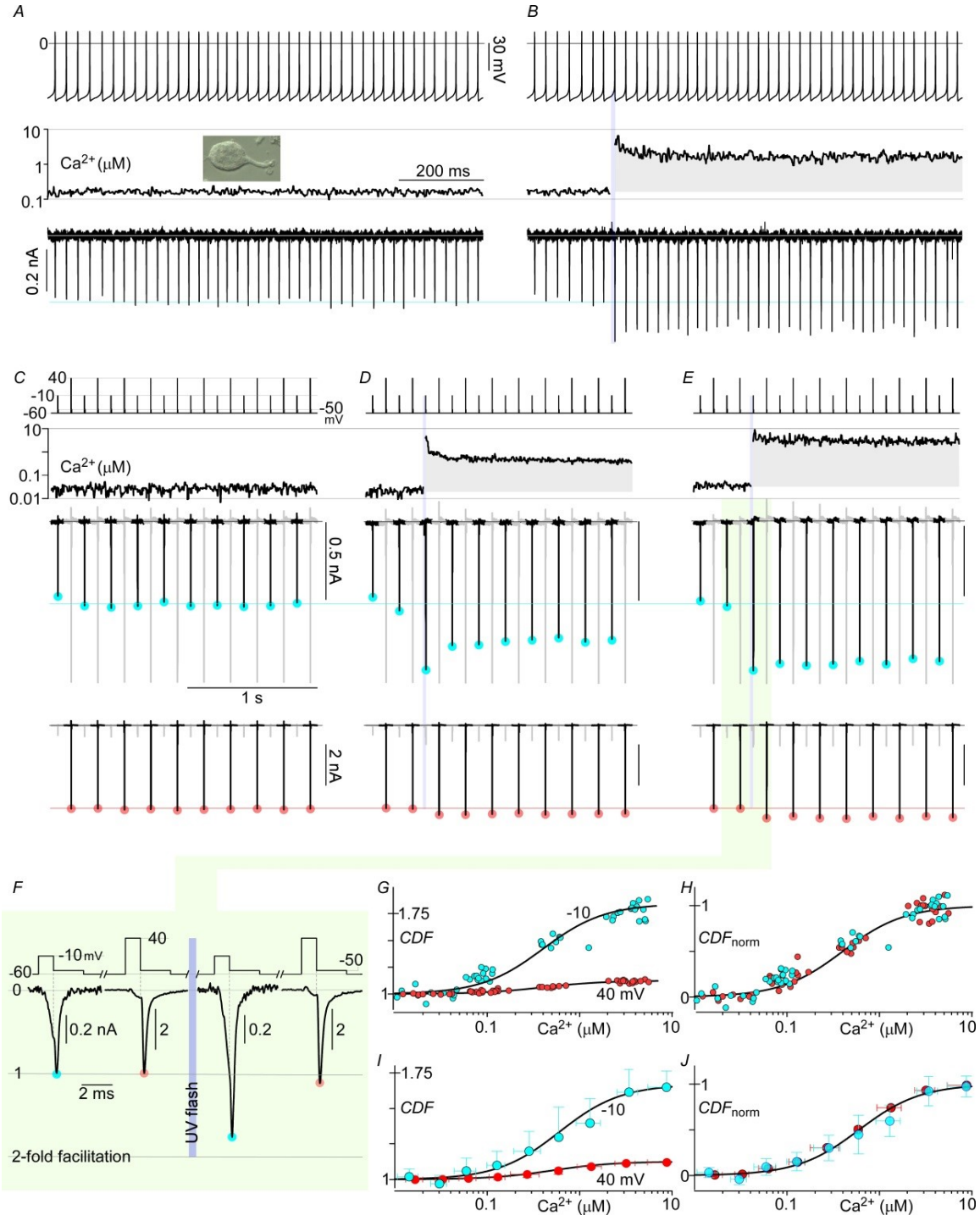
A
B
0
30 mV
10
1
0.1
Ca2+ (μM)
200 ms
0.2 nA
C
D
E
40
-10
-60
-50
mV
10
0.1
0.01
Ca2+ (μM)
0.5 nA
1 s
2 nA
F
-10 mV
40
-50
-60
0
0.2 nA
2
0.2
2
1
2 ms
UV flash
2-fold facilitation
G
1.75
CDF
-10
40 mV
0.1
Ca2+ (μM)
10
H
1
CDFnorm
0
0.1
Ca2+ (μM)
10
I
1.75
CDF
-10
40 mV
1
0.1
Ca2+ (μM)
10
J
1
CDFnorm
0
0.1
Ca2+ (μM)
10

Figure 11

Figure 11. CDF in native $Ca_V2.1$ channels within cerebellar Purkinje cells.

A, acutely-dissociated Purkinje cells were identified by their large size and tear-drop shape (picture *inset*). $Ca_V2.1$ Ca^{2+} currents stimulated by ~50Hz action potential waveforms are stable and show little facilitation near baseline Ca^{2+}. *B*, Ca^{2+} uncaging in the midst of the AP train triggers large CDF. *C*, Ca^{2+} sensitivities of CDF at 2 voltages were determined with a protocol similar to that in Figure 10C. Steady currents are seen in the absence of uncaging (*left*), while uncaging to different Ca^{2+} levels trigger large increases in current (*middle* and *right*). *F*, tail currents around the time of Ca^{2+} uncaging are shown with expanded time-base, normalized to the peak levels prior to uncaging. *G*, relative increase in peak tail currents plotted against Ca^{2+} for the above exemplar cell yields steady-state *CDF vs*. Ca^{2+} curves for -10 and 40 mV. *H*, data in *G* normalized demonstrates the same Ca^{2+} sensitivity for the two voltages. *I-J*, population data (mean±STD) shows similar results, with $n = 5$ cells. Fit curves have $n_{Hill} = 1.3$, $EC_{50} = 0.5$ μM.

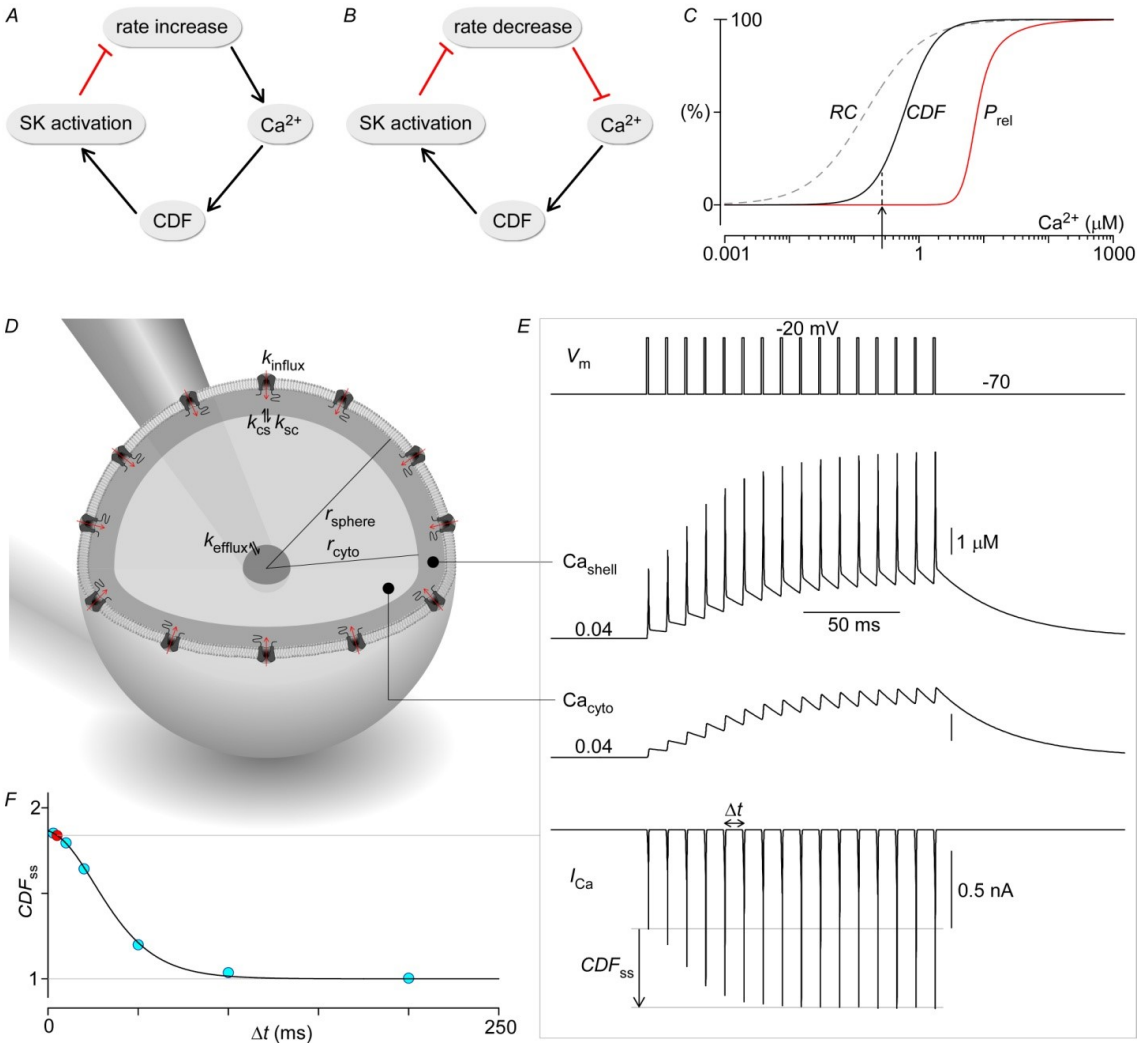

Figure 12

Figure 12. Neurophysiological implications of Cav2.1 CDF

A, Diagram illustrating the role of CDF in homeostatic rate control of Purkinje cells. Increased rate of Purkinje cell spiking elevates global Ca^{2+}, which shifts facilitates Cav2.1 opening via CDF. Increased Cav2.1 activity in turn activates colocalized SK channels, resulting in stronger membrane hyperpolarization that then reduces spiking rate. *B*, Similarly, decreased spiking rate leads to lower global Ca^{2+}, reduced CDF, and less SK activation, finally resulting in weaker hyperpolarization and restoration of spiking rate. *C*, Ca^{2+} dose response curves for CDF (black) and neurotransmitter secretion (red). Additionally, Ca^{2+} sensitivity of paired-pulse facilitation due to residual Ca^{2+} (Regehr *et al.* 1994) (dashed line) shows an additional mechanism specifying the extent of transmitter release. *D*, Generic nerve terminal is modeled as a sphere with Cav2.1 channels uniformly expressed on the membrane. A thin shell compartment adjacent to the membrane forms the nanodomain within which the CDF Ca^{2+} sensor CaM, which acts as

a channel subunit, actually senses Ca^{2+}. Ca^{2+} within the shell compartment can diffuse to and from the much larger cytosolic compartment. The cytosolic compartment is in turn connected to a third and much larger compartment representing the axon branch itself. The Ca2+ concentration within this axonal compartment is set at a fixed resting level, and thus serves to returns Ca^{2+} within the terminal back to its resting levels after periods of activity. The parameters for this system are: r_{sphere} = 8 µm, r_{cyto} = 7.9 µm, N_{chan} = 60000, $k_{sc} = k_{cs} = 6x10^{-13}$ $dm^3 \cdot ms^{-1}$, and $k_{efflux} = 6x10^{-14}$ $dm^3 \cdot ms^{-1}$, with the parameters are adjusted such that simulations in *E* match data from Cuttle *et al.* (1998). Note that the simulated volume is larger than physical terminals because Ca^{2+} buffers are not explicitly modeled here, for simplicity. *E*, simulations of the system in *D* in response to 100-Hz step depolarizations to -20 mV. Cav2.1 currents are elicited (bottom row) by the voltage stimulation (top row), which results in a build-up of Ca^{2+} in both the shell (second row) and cytosolic (third row) compartments. The increased Ca^{2+} induces CDF of Cav2.1 channels, resulting in gradual amplification of I_{Ca}, Ca_{shell} and Ca_{cyto}. *F*, CDF at steady state is inversely related to stimulation frequency (blue dots and black line), largely due to frequency-dependent accumulation of residual Ca^{2+} (100 Hz exemplar in red).

Part 2: Uncovering aberrant mutant PKA function with flow cytometric FRET

Introduction

Interactions between proteins underlie much of the richness and complexity of biology. Proteins rarely act in isolation; rather, their function is often modulated, regulated or localized by their interactions with other proteins [80]. Many biological processes, from signal transduction to synaptic vesicle release, require the sequential binding and unbinding of a network of proteins to achieve their intended effects. Consequently, techniques for detecting and characterizing protein-protein interactions (PPI) have been instrumental in advancing our understanding of biology.

These methods are broadly divided into techniques that probe protein-protein interactions under extracellular or intracellular conditions. The former group includes biochemical assays, surface plasmon resonance (SPR) and isothermal calorimetry (ITC), which are generally highly quantitative, but are relatively low-throughput and can be limited by the demands of isolating and purifying sufficient quantities of high-quality protein. The latter group includes techniques like two-hybrid assays, which are higher-throughput and have the advantage of interrogating proteins within an intracellular milieu, but are less quantitative with higher rates of false-positives and false-negatives [81]. More recently, Förster resonance energy transfer (FRET)-based methods have gained popularity [82], as they combine the ease and biological relevance of *in-cell* approaches with the specificity and quantitative power of biochemical assays.

For instance, quantitative FRET has been used successfully to characterize PPI through the construction of live-cell FRET-based binding curves [83, 84]. Imagine a scenario where proteins *A* and *B* bind to form *AB* with dissociation constant K_d, as illustrated in **Figure 13A**. Traditional approaches to characterize their binding would do

so by manually titrating one species against a fixed concentration of the other whilst measuring the amount of bound product formed. In the live-cell FRET-based approach, proteins *A* and *B* are first tagged with fluorescent proteins that have high FRET potential, such as the GFP-variants Cerulean [85] (Cer) and Venus [86] (Ven), and expressed in live cells. The stochastic expression of *A* and *B* in different cells then results in a natural titration of the two species, with the genetically-tagged Ven and Cer reporting the concentrations of *A*, *B* and *AB* within each cell through their direct fluorescence and the amount of FRET. With an appropriate binding model, a binding curve can then be constructed, with each cell contributing a point on the graph (**Figure 13B**). As a result, a good estimation of binding affinity typically requires interrogation of a large number of cells expressing a wide range of concentrations around the K_d. Prior attempts to apply this methodology utilized wide-field fluorescence microscopy to measure fluorescences and FRET efficiencies cell-by-cell [83, 84]. Although this approach enabled FRET-based binding studies to be done on most fluorescence microscopes, obtaining each binding curve was a slow and laborious task where fluorescent cells had to be measured one at a time. Moreover, the microscope required frequent and onerous calibrations to offset instrument drift. These challenges placed limits on the scope of problems addressable by this method.

The advent of the flow cytometer, an instrument capable of performing multichannel fluorescence measurements of single cells in a high-throughput manner, raised hopes that FRET-based protein interaction studies could be done more simply and efficiently. However, quantifying FRET with flow cytometry is challenging—acceptor

photobleaching [87] is impractical given the sub-millisecond excitation times, while flow cytometers capable of fluorescence lifetime measurements [88, 89] are currently not widely available. Spectral methods for FRET determination predominate [90, 91], though most groups only estimate FRET semi-quantitatively, tracking relative increases in donor quenching or sensitized acceptor emission, using this as a high-throughput approach to detect PPI's [92, 93]. However, semi-quantitative FRET measurements are easily distorted by relative donor and acceptor expression levels [94], potentially leading to erroneous estimations of binding. Moreover, proteins that bind weakly but have a favorable FRET conformation (e.g. aligned dipoles, shorter bound distance) might have a similar FRET efficiency to proteins that bind strongly albeit with unfavorable FRET conformations. Thus, the capability to measure binding in a dose-dependent, high-throughput manner is important, as it not only results in higher confidence of detecting actual binding, but also enables large scale studies of protein binding as with drug design and discovery, and structure-function analyses.

In this study, we describe a simple method to calibrate a commercially-available flow cytometer to perform quantitative FRET, using this approach both to construct FRET-based binding curves of fluorescent protein-tagged proteins and to exploit a FRET-based activity sensor. We validate our FRET-based binding assay by comparing our binding affinity estimates with those obtained from ITC, and use our method to discern the mechanism by which a recently discovered point mutation (L206R) in the PKA catalytic subunit results in adrenal Cushing's syndrome. Our results demonstrate the

ease, efficiency and broad utility of flow cytometric FRET, raising expectations that FRET-based binding assays will contribute to a new era of biological discovery.

Methods

Molecular biology

All cDNA plasmid constructs used in this study have transcription driven by the CMV promoter. Plasmids encoding monomeric Cerulean (Cer-C1) and Venus (Ven-C1) as well as dimers C32V, C40V, C50V and CTV were gifts from Steven Vogel (NIH). Cer-N3, Ven-N3 and mCherry-N3, with multiple cloning sites N-terminal to the fluorescent proteins (FP), were generated through PCR amplification of the FPs (startless) using primers P01/P02 (primers listed in **Table 1**) from Cer-C1, Ven-C1 and mCherry-C1 (Clontech) and insertion into EGFP-N3 (Clontech) with BamHI/BsrGI. These C1 and N3 plasmids serve as the backbones to which other proteins can be fused to Cer or Ven at either terminus. C22V and C22R were generated by cutting and inserting from Ven-N3 or mCherry-N3 into Cer-C1 with BglII/XbaI. The carboxy-terminal SH3 domain from Mona/Gads was PCR amplified from mouse tail DNA with P03/P04, and inserted into Ven-C1 with BglII/HindIII. Its various Cer-tagged 13 amino-acid binding partners were synthesized by annealing, cutting and inserting complementary DNA primer pairs (P05-P28) bearing the desired sequence into Cer-C1 with XhoI/HindIII. Plasmids containing human PKA_{cat} (pDONR223-PRKACA), PKA_{reg} (pDONR223-PRKAR1A and pDONR223-PRKAR2B) and PKI (pRSV-PKIv2) were obtained from addgene.org, while AKAR4 was a gift from Jin Zhang (Hopkins). PKA_{cat} was PCR amplified with P29/P30 and cut with BamHI/HindIII before insertion into Cer-C1, Ven-C1 and mCherry-C1 using BglII/HindIII to make Cer-PKA_{cat}, Ven-PKA_{cat} and mCherry-PKA_{cat}. The L206R

mutation in PKA_{cat} was made using overlap-extension PCR with overlap primers P31/P32 and flank primers P29/P30, with reinsertion using EcoRV/XhoI. PKA_{reg}-Cer and PKA_{reg}-Ven were made by PCR amplification of PKA_{reg} R1α (P33/P34) or PKA_{reg} R2β (P35/P36) and insertion into Cer-N3 and Ven-N3 with KpnI/BamHI or KpnI/BamHI/BglII respectively. PKI-Cer, PKI-Ven and PKI-mCherry were constructed by PCR amplification using P37/P38 and insertion into Cer-N3, Ven-N3 or mCherry-N3 using KpnI/BamHI. Untagged PKA_{cat} was made by PCR amplification using P39/P40 and insertion into pcDNA3.1 (Invitrogen) using KpnI/XbaI, while untagged PKA_{reg} (R1α) was made by cutting PKA_{reg}-Cer (R1α) and inserting into Cer-C1 with KpnI/BamHI. All constructs were verified by restriction digest and Sanger sequencing.

Cell culture, transfection and preparation for flow cytometry

HEK293 and HEK293T cells were cultured using sterile techniques on 6 well plates at 37°C with 5% CO_2. Our culture media contained Dulbecco's modified Eagles' medium (DMEM; Invitrogen) supplemented with 10% FBS, 5 mM L-glutamine, 50 units/ml penicillin and 50 µg/ml gentamicin. Cells within each well were transiently transfected at 75% confluency using polyethylenimine (Polysciences) with a 3:1 (PEI:DNA) ratio. 16-48 hours after transfection, cells were trypsinized and transferred to 1.6 ml microcentrifuge tubes where they were washed twice and then resuspended in 500 µl of HEPES-buffered Tyrode's solution. Cells were then transferred to 5 mL round bottom polystyrene tubes for use in the flow cytometer. In most experiments, 100 µM cycloheximide was added to cells 2-4 hours prior to flow cytometric interrogation to halt new protein synthesis, allowing immature fluorescent proteins to fully mature.

Flow cytometry

We used an Attune® acoustic focusing flow cytometer (Life Technologies) equipped with violet (405 nm) and blue (488 nm) lasers. Forward and side scatter signals were developed from the violet laser and used to gate for single, healthy cells. S_{Cer}, and S_{FRET} channels were taken from the violet laser, while S_{FRET} and S_{red} were off the blue laser. Optical filters used for S_{Cer}, S_{Ven}, S_{FRET} and S_{red} were 480/50, 530/30, 530/30 and 600LP respectively. Flow cytometric signals were collected using "high-sensitivity" mode at a flow rate of 100 µl/min. Cells are illuminated for 40 µs by each of the two lasers, with a time-of-flight of 1.2 ms from the violet to the blue laser. We confirmed that there was no appreciable photoactivation of Venus by violet (405nm) light (data not shown). Maintenance and performance tracking of the flow cytometer were performed with Attune® performance tracking beads (Life Technologies) in accordance with manufacturer's specifications. FITC beads used for absolute calibration and for calibration across PMT voltages were Quantum™ FITC-5 MESF (Bangs Laboratories, Inc). Fluorescence data was exported as FCS files for further processing with custom MATLAB® (Mathworks) software.

Several control experiments were prepared every time the flow cytometer was used: cells exposed to PEI but without plasmids served as our blank controls, and used to define the background level for each channel (BG_{Cer}, BG_{Ven}, BG_{FRET}, BG_{Red}); Cer, Ven and mCherry each expressed by itself gives the proportionality constants (R_{D1}, R_{D2}, R_{D3}, R_{A1}, R_{A2}, R_C) for spectral unmixing; Cer and Ven expressed together provides an estimate of the concentration-dependent collisional FRET (**Figure 15M-R**). The series of Cer-Ven dimers for g_A/g_D and f_A/f_D calibration were performed monthly, while calibration with

Quantum™ beads were used only when the flow cytometer has drifted significantly from baseline, as determined by serial performance testing.

Data processing

Our custom software serves 3 functions: 1) as a means to organize experimental constructs by name, type and date, 2) to determine and store the necessary instrument- and fluorophore-dependent constants, and 3) to transform the raw fluorescence values into FRET efficiencies and binding curves, as outlined in **Figure 13F**. For all samples, single cells are first selected through sequential gating with forward- and side-scatter plots (**Figure 14A-C**). Background levels are determined from blank cells by taking the mean fluorescence levels of each channel after discarding 1% of outliers, and used for background subtraction in all subsequent analysis (**Figure 14D-E**). R_{A1} and R_{A2} were taken from Ven-only cells (**Figure 15A**) by taking the slope of a linear regression of S_{FRET} *vs* S_{Ven}, and S_{FRET} *vs* S_{Red} (**Figure 15C-D**). Similarly, R_{D1}, R_{D2} and R_{D3} are determined Cer-only cells with S_{FRET} *vs* S_{Cer}, S_{Ven} *vs* S_{Cer}, and S_{Red} *vs* S_{Cer} respectively (**Figure 15E-H**). Importantly, these relations between channels become supralinear at very high fluorescence intensities (**Figure 15B**), likely due to inner filter effects, and so highly expressing cells were excluded from analysis. Moreover, to minimize the influence of outliers, binned median fluorescences were used to calculate the linear regressions from which the proportionality constants (R_{D1}, R_{D2}, R_{D3}, R_{A1}, R_{A2}) are determined. The unmixed signals (Cer_{direct}, Ven_{direct}, Ven_{FRET} and *Cherry*) were then obtained by inverting the following relation:

$$\begin{bmatrix} S_{Cer} \\ S_{Ven} \\ S_{FRET} \\ S_{Red} \end{bmatrix} = \begin{bmatrix} 1/R_{D1} & 0 & 0 & 0 \\ R_{D2}/R_{D1} & 1/R_{A1} & 0 & 0 \\ 1 & 1 & 1 & 0 \\ R_{D3}/R_{D1} & 1/R_{A2} & 0 & 1 \end{bmatrix} \begin{bmatrix} Cer_{direct} \\ Ven_{direct} \\ Ven_{FRET} \\ Cherry \end{bmatrix}$$

These unmixed quantities allow calibration of g_A/g_D and f_A/f_D using a series of Cer-Ven dimers and equation (6), calculation of mean cellular FRET efficiencies according to equations (3) or (4), as well as determination of N_{Ven} and N_{Cer} using equations (1) and (1). As detailed in the main text, the absolute brightness of Ven (g_Af_A) was determined via comparison with FITC beads, and the brightness of Cer (g_Df_D) assessed relative to that of Ven through the Cer-Ven dimer series. We also estimated the brightness of mCherry ($g_{Ch}f_{Ch}$) relative to Cer using a Cer-mCherry dimer (**Figure 15L**), which enabled us to calculate N_{Cherry} from *Cherry*. It is especially important to determine experimentally the brightness of Cer relative to that of Ven because an accurate gauge of the brightness ratio is so crucial to calculating the FRET efficiency. Finally, we estimated the volume of an average HEK293 cell by pipetting trypsinized HEK293 cells onto a glass coverslip, and measuring the radii of these loosely attached spherical cells with confocal microscopy (**Figure 14F**).

Results

Quantitative FRET with flow cytometry

With proper calibration and selection of fluorescent molecules, fluorescence signals from the flow cytometer can reveal the concentrations of each fluorescent species and the average FRET efficiency within each cell. We chose for our experiments Cer and Ven bearing monomeric A206K mutations [95] because of its favorable FRET properties, and compatibility with a dual violet and blue laser-equipped flow cytometer. A brief outline

of our workflow is outlined in **Figure 13C-F**: HEK293(T) cells within 6 well plates are transfected with constructs containing Cer and/or Ven, and harvested 16-48 hours after transfection into round bottom tubes. Each round bottom tube is sequentially loaded onto the flow cytometer, where cells are focused into single stream and interrogated one-at-a-time by the two lasers. The resultant fluorescence and laser scatter are optically filtered into multiple channels, and detected, amplified and digitized. The recorded data is then analyzed using custom MATLAB code according to the sequence diagrammed in **Figure 13F**. Fluorescence signals from single cells (**Figure 14**) are converted through linear unmixing (see **Methods** and **Figure 15**) into the three fundamental signals of the Cer/Ven system—direct Cer emission (Cer_{direct}), direct Ven emission (Ven_{direct}), and sensitized Ven emission from FRET (Ven_{FRET}). Ven_{direct} is directly proportional to the number of Ven molecules (N_{Ven}), while Cer_{direct} is proportional to the number of Cer molecules (N_{Cer}) not quenched by FRET as follows,

$$Ven_{direct} = g_{Ven} f_{Ven} N_{Ven} \quad (1)$$

$$Cer_{direct} = g_{Cer} f_{Cer} N_{Cer} (1 - \langle E \rangle) \quad (2)$$

where $g = \gamma\epsilon$ with γ encompassing the instrument-dependent, and ϵ the fluorophore-dependent, aspects of fluorophore excitation, including laser power, attenuation by optical components, and the extinction coefficient at the laser wavelength. $f = \xi\phi$ with ϕ representing the fluorophore-dependent, and ξ the instrument-dependent, terms of fluorophore emission, such as the quantum efficiency, optical filtering, as well as linear photodetection, amplification and digitization of fluorescence. N_{Ven} can be determined once the brightness of a single Ven molecule ($g_{Ven} f_{Ven}$) is known, but determination of

N_{Cer} further requires an estimation of the average FRET efficiency $\langle E \rangle$. The average FRET efficiency is a function of *g* and *f*, and depending on whether it is measured with respect to the donor (Cer) or acceptor (Ven), can be expressed as (**Appendix IIA**. Quantitative FRET with flow cytometry):

$$\langle E \rangle_{Ven} = \frac{g_{Ven}}{g_{Cer}} \frac{Ven_{FRET}}{Ven_{direct}} \quad (3)$$

$$\langle E \rangle_{Cer} = \frac{Ven_{FRET}}{Ven_{FRET} + \frac{f_{Ven}}{f_{Cer}} \cdot Cer_{direct}} \quad (4)$$

Thus, absolute calibration of the flow cytometer depends on determination of *g* and *f* for both Ven and Cer. Once obtained, total concentrations of Cer (C_{Cer}) and Ven (C_{Ven}) are calculated from N_{Cer} and N_{Ven} after accounting for cellular volume (**Figure 14F**). Achieving precise accounting of the fluorescent species expressed in live cells then allows the construction of FRET-based binding curves.

Absolute calibration of the flow cytometer

Absolute calibration is achieved when fluorescence signals can be converted to an absolute number/concentration of fluorescent proteins with equations (1) and (1). Whereas the instrument-dependent terms γ and ξ are difficult to determine, the fluorophore-dependent terms ϵ and ϕ are readily accessible [85, 86]. Fortunately, calibration standards for flow cytometry with known numbers of fluorochromes are commercially available, allowing determination of γ and ξ. We obtained beads conjugated with predetermined numbers of FITC molecules, as FITC is spectrally close to Venus. We can then absolutely calibrate our flow cytometer to the brightness of a single FITC molecule ($g_{FITC} f_{FITC}$) from the slope of the fluorescence *vs* number of FITC

molecules relation (Figure 2A). Since the instrument is unchanged, γ and ξ are constant, and we can directly calculate for the brightness of a single Ven using known values of ϵ and ϕ for Ven and FITC [96, 97]:

$$g_{Ven}f_{Ven} = g_{FITC}f_{FITC}\frac{\epsilon_{Ven}\phi_{Ven}}{\epsilon_{FITC}\phi_{FITC}} \quad (5)$$

As seen with equations (3) and (4), calculating average FRET efficiencies requires the ratios of *g* and/or *f* for both Cer and Ven. These absorption and emission ratios can be determined [98, 99] with the following rationale: with Cer-Ven fusion proteins, the FRET efficiency should be the same regardless of the perspective from which it is measured, as long as there is one acceptor for every donor molecule (**Appendix IIA**). That is, since $\langle E\rangle_{Cer} = \langle E\rangle_{Ven}$ in this scenario, equations (3) and (4) can be rearranged to obtain:

$$\frac{Ven_{FRET}}{Cer_{direct}} = \frac{g_{Cer}}{g_{Ven}}\frac{Ven_{direct}}{Cer_{direct}} - \frac{f_{Ven}}{f_{Cer}} \quad (6)$$

Equation (6) predicts a linear relationship from which the slope and intercept yields our desired ratios (**Figure 16A**). Plotting the appropriate quantities for five Cer-Ven dimers with varying linker lengths (from 22-228 amino acids), we initially obtained scatter plots with significant tails towards the origin (**Figure 17A**). Upon further examination, we found that $\langle E\rangle_{Cer}$, but not $\langle E\rangle_{Ven}$, had large variance (**Figure 17B**,**C**). We hypothesized that this asymmetric result was likely due to the presence of a significant fraction of non-fluorescent immature Ven molecules. We thus applied the protein synthesis inhibitor

cycloheximide for 2 hours prior to flow cytometric interrogation, reasoning that we could reduce the immature fraction by halting new synthesis of fluorescent proteins and allowing existing Ven molecules to fully mature. Indeed, doing so dramatically reduced the variance of $\langle E \rangle_{Cer}$, resulting in the graphs shown in **Figure 16.** The plot in **Figure 16**B enabled us to obtain estimates of 0.0178 and 1.65 for g_{Ven}/g_{Cer} and f_{Ven}/f_{Cer}. These quantities when multiplied also give the relative brightness of Cer and Ven, thus furnishing sufficient constraints to achieve absolute calibration. We tested our estimates of the absorption and emission ratios both by examining the $N_{\mathrm{Cer}}/N_{\mathrm{Ven}}$ ratio, and by plotting $\langle E \rangle_{Cer}$ and $\langle E \rangle_{Ven}$, of these dimers. Reassuringly, $N_{\mathrm{Cer}}/N_{\mathrm{Ven}}$ is near 1 (**Figure 16C**), and $\langle E \rangle_{Cer} = \langle E \rangle_{Ven}$ for each dimer, with dimers bearing shorter linker lengths having higher FRET efficiencies (**Figure 16D**).

Live cell FRET-based binding curves

We then sought to construct FRET-based binding curves with a test set of peptides whose binding had previously been well-characterized by ITC [100]. We created constructs tagged with either Cer or Ven (**Figure 18A**) using the exact sequence of their peptides. After expressing them in HEK293 cells, we obtained single cell measurements of C_{Cer}, C_{Ven} and $\langle E \rangle$ using flow cytometry. Because proteins that bind to one another do so in hand-and-glove fashion, the bound complex adopts a conformation with FRET efficiency E_{max}. Since unbound proteins have negligible FRET,

$$\langle E \rangle_{Ven} = A_b E_{max} \tag{7}$$

$$\langle E \rangle_{Cer} = D_b E_{max} \tag{8}$$

where A_{b} and D_{b} refer to the fraction of acceptor (Ven) and donor (Cer) molecules bound. Furthermore, the 1:1 binding model has the following analytic solution:

$$A_b = \frac{(C_{Cer} + C_{Ven} + K_d) - \sqrt{(C_{Cer} + C_{Ven} + K_d)^2 - 4C_{Ven}C_{Cer}}}{2C_{Ven}} \quad (9)$$

$$D_b = \frac{(C_{Cer} + C_{Ven} + K_d) - \sqrt{(C_{Cer} + C_{Ven} + K_d)^2 - 4C_{Ven}C_{Cer}}}{2C_{Cer}} \quad (10)$$

Thus, our approach is to iteratively minimize the difference between A_b as calculated using equations (7) and (9), or D_b using equations (8) and (10), with the parameters E_{max} and K_d (**Figure 19**). Binding curves of Ven-SH3 (Gads, aa.256-322) with five binding partners (SLP-76, aa.231-243; Gab1, aa.515-527; USP8, aa.403-415; SLP-76 (D236K), aa.231-243; SLP-76 (K240R), aa.231-243), are presented in **Figure 18B**. Each binding curve is constructed from a single trial where >100,000 fluorescent cells were collected, with each cell contributing a dot on the plot. With each trial taking 3-5 minutes, data for all 5 constructs were collected in less than 30 minutes. Data from this large number of cells form a cloud charting the course of each binding curve with high resolution. Protein pairs with binding affinities within an order of magnitude of 1 μM can be maximally resolved, with points decorating the entirety of the binding curve (**Figure 18B**, green and blue). On the other hand, binding curves of weakly-binding (**Figure 18B**, cyan and red) or strongly-binding (**Figure 18B**, black) protein pairs would miss points at both extremes (A_b or D_b near 0 or 1), because of the difficulty of expressing proteins at concentrations upwards of several hundred μM in live cells, or because of the difficulty of quantifying faintly fluorescent cells when proteins are weakly expressed. Nonetheless, binding curves that capture a significant fraction of the full range of binding can still yield good estimates of binding affinities through the minimization procedure described above (**Figure 19**). Analyzing binding data from all peptide pairs assessed, we obtained a

relation between affinity estimates using flow cytometry and those using ITC (**Figure 18C**). The results impressively demonstrate that affinities estimated with both methods are highly correlated, though estimations with flow cytometry yielded affinities that were on average ~3-fold lower than those with ITC (**Figure 18C**, red line). The discrepancies between these two methods might be the result of small alterations in the energetics of binding due to the steric influence of the comparatively large fluorescent protein tags. They might also reflect the presence of invisible inhibitors of the binding reaction, due to endogenous inhibitor proteins and/or non-fluorescent immature fluorophores. Regardless of the cause, these results show that high-resolution FRET binding curves can be constructed quickly and efficiently with flow cytometry, yielding estimates of binding affinity comparable to those by *in vitro* studies.

Regulation of Protein Kinase A—implications of a catalytic subunit mutant (L206R)

Recently, a number of papers were published describing the remarkable discovery that a large subset of patients with adrenal Cushing's syndrome harbored the same point mutation (L206R) in the P+1 loop of the catalytic subunit of PKA (PKA_{cat}) [101-105]. This mutation was found to increase baseline PKA activity, with this gain of function largely thought to be due to disruption of binding between PKA_{cat} and its regulatory subunit (PKA_{reg}). However, there remained doubt as to whether some residual binding was preserved, given that PKI, whose binding interface with PKA_{cat} structurally resembles PKA_{reg}, is still able to bind to and inhibit the mutant PKA_{cat} [102]. We thus sought to resolve this question, using flow cytometric FRET to perform both binding and

functional studies to understand how the L206R mutation affects PKA_{cat}'s ability to interact with its various partners and thus alter its function.

We first tagged PKA_{cat} and PKA_{reg} with Ven and Cer, and performed our flow cytometric live cell FRET binding assay as before. In this case however, a 2:2 binding model (**Appendix IIB**) was used because the PKA holoenzyme forms a tetrameric complex [106]. The results indicate clear binding between the two wild-type subunits, with cells lining the binding curve (**Figure 20A**). The binding affinity was lower than anticipated [107], likely because of the abundance of endogenous PKA in HEK293 cells. Nevertheless, flow cytometric analysis of the mutant L206R catalytic subunit demonstrated complete disruption of binding with the R1α regulatory domain throughout the entire range of expression (**Figure 20B**). Experiments with the R2β regulatory subunit yielded similar results (**Figure 21A-B**). We then assessed binding between PKA_{cat} and its natural inhibitor PKI. A FRET binding study of PKI-Ven with the wild-type Cer-PKA_{cat} was obtained (**Figure 20C**), which showed clear binding (K_d = 3.5 μM), though again with lower affinity than *in vitro* values[108] due to competition from endogenous proteins. Assessing binding of PKI with L206R PKA_{cat} (**Figure 20D**), we found surprisingly that binding was preserved, though with weaker affinity (K_d = 10 μM). Thus, these experiments show that the interfacial L206R mutation differentially affects PKA_{cat}'s interactions with two of its natural binding partners.

The partner-specific effects of L206R on PKA_{cat} binding suggested that the rate of phosphorylation of its targets might also be affected, as transient binding of PKA_{cat} to its substrate is a prerequisite to catalysis. We thus sought to examine the functional

consequence of this mutant PKA_{cat} using the FRET-based PKA activity sensor AKAR4 [109]. **Figure 22A** diagrams a kinetic scheme of AKAR4, with AKAR4 phosphorylation requiring an intermediate enzyme-substrate bound state reached through dissociation constant k_{-1}/k_1 and departed through catalytic rate constant k_2. Dephosphorylation by endogenous phosphatases is represented by a single rate constant k_3. Because AKAR4 exists primarily in 2 states—unphosphorylated (E_{min}) or phosphorylated (E_{max}), the average FRET efficiency of a cell expressing AKAR4 is linearly related to the fraction of AKAR4 phosphorylated (P) by the following equation:

$$\langle E \rangle = P \cdot E_{max} + (1 - P) \cdot E_{min} \tag{11}$$

E_{min} can be approximated by expressing AKAR4 in live cells, and applying the PKA inhibitor H-89 (thus driving k_2 to 0) whilst measuring FRET with flow cytometry (**Figure 22B**, black). On the other hand, we can get close to E_{max} by over-expressing PKA_{cat} (**Figure 22B**, red). Surprisingly, $\langle E \rangle$, which should be independent of substrate concentration (equation 11), appeared to increase with AKAR4 expression. We reasoned this was likely because cells expressing more AKAR4 tend to have higher PKA_{cat} expression as well. Indeed, PKA_{cat} levels should influence P with the following relation (**Appendix IIC**):

$$P = \frac{\langle E \rangle - E_{min}}{E_{max} - E_{min}} = \frac{P_{max} \cdot PKA_{cat}}{K_e + PKA_{cat}} \tag{12}$$

where $P_{max} = k_2/(k_2 + k_3)$, $K_e = K_m(1 - P_{max})$, and $K_m = (k_{-1} + k_2)/k_1$. We estimated k_3 with flow cytometry by examining the time-dependent decay of the mean population $\overline{\langle E \rangle}$ of cells co-expressing AKAR4 and PKA_{cat} after exposure to 100 μM H-89

(**Figure 22C**). We examined cells expressing >4 μM AKAR4 to ensure all cells probed started with saturating PKA activity, and estimated from the exponential decay that $k_3 \sim 0.03$ sec^{-1}.

Equation (2) suggested a revealing relationship between AKAR4 phosphorylation and PKA_{cat}, and we thus sought to acquire a means of tracking cellular PKA_{cat} levels, while continuing to monitor PKA activity with Cer-Ven FRET. We found that fully-matured mCherry fulfilled this role beautifully, as it was excitable with the 488 nm laser with insignificant bleedthrough into other channels (**Figure 15I-K**). Additionally, mCherry levels can be estimated after accounting for Ven spectral overlap, and single fluorophore brightness calibration with a Cer-mCherry dimer (**Figure 15L**). Having added additional bandwidth to our flow cytometric system, we first co-expressed in live cells mCherry-tagged PKI with AKAR4 and obtained a largely flat FRET vs mCherry-PKI curve, where we find that increasing expression of mCherry-PKI depressed $\langle E \rangle$ to E_{min} (**Figure 22D**, black). On the other hand, co-expression of mCherry-tagged PKA_{cat} and AKAR4 yielded a curve (**Figure 22D**, red) that satisfies equation (2) with $y_{max} = 0.445$ and $K_e = 0.1$ μM. A similar experiment with mCherry-tagged mutant (L206R) PKA_{cat} instead yielded a markedly different curve (**Figure 22D**, green) with $y_{max,mut} = 0.385$ and $K_{e,mut} = 12$ μM. The reduction in $y_{max,mut}$ and 100-fold increase in $K_{e,mut}$ suggests that the L206R mutation not only weakens the catalytic subunit's interaction with AKAR4 ($k_{1,mut} << k_1$), but also impairs catalysis ($k_{2,mut} < k_2$). Since $k_2 >> k_{-1}$, our estimates of k_3 allow us to approximate $k_1 \approx k_3/K_e = 1\text{x}10^5$ $M^{-1}sec^{-1}$ and $k_{1,mut} \approx 0.8\text{x}10^3$ $M^{-1}sec^{-1}$. On the other hand, precise quantification of k_2 is more difficult, with E_{max} too

close to y_{max} ($P_{max} \sim 1$) to be resolved. Nonetheless, based on prior biochemical studies [110] we can assume $k_2 \sim 20$ sec^{-1}, which yields: $k_{2,mut} \sim 0.1$ sec^{-1}.

Addition of an "inhibitor", such as PKA_{reg} or PKI, to the PKA_{cat} + AKAR4 system results in a scheme diagrammed in **Figure 22E**. The inhibitor competitively binds to PKA_{cat}, effectively reducing the number of catalytic molecules that can phosphorylate AKAR4. The solution to this system reveals an interesting feature of the PKA regulatory system (**Figure 22F**)—the fraction of AKAR4 phosphorylated depends not only on the relative number of catalytic and regulatory subunits (C_{tot}/I_{tot}), but also on the relative rates of PKA_{reg}-PKA_{cat} binding and AKAR4 cycling (K_I/K_e) (**Supp. D**). It is intuitive that $C_{tot}/I_{tot} < 1$ is required for baseline suppression of PKA_{cat} activity since there has to be at least one regulatory subunit for every catalytic subunit. On the other hand, K_I/K_e reflects the competition between PKA_{reg} and substrate for PKA_{cat}, describing the predilection of PKA_{cat} for either participating in the phosphorylation or quiescent pathway. When $K_I/K_e << 1$ (**Figure 22F**, red), PKA_{cat} strongly prefers PKA_{reg}, and so is mostly inactive even when C_{tot}/I_{tot} is just under 1. On the other hand, if $K_I/K_e >> 1$ (**Figure 22F**, green), as would be the case with PKA_{cat} and $(cAMP)_2$-PKA_{reg}, PKA_{cat} is mostly active, even when outnumbered by regulatory subunits. Furthermore, this model predicts that examining AKAR4 phosphorylation when both PKA_{cat} and R1α are expressed in HEK293 cells would yield data that allows interpretation of the relative binding affinities of PKA_{cat}. For instance, if $K_I/K_e << 1$, we would expect a bimodal FRET distribution with most cells displaying either high or low FRET, since the steepness of the P vs C_{tot}/I_{tot} curve (**Figure 22F**, red) indicates that only a small fraction of cells would lie within the narrow range of

C_{tot}/I_{tot} necessary for intermediate AKAR4 activity. Indeed, when we transfect untagged PKA_{cat} and R1α with AKAR4, we observe a bimodal distribution of FRET activity (**Figure 22G**), suggesting that $K_I/K_e << 1$. Similarly, co-expression of PKI with wild-type PKA_{cat} yields a bimodal FRET distribution (**Figure 21C**), again consistent with $K_I/K_e << 1$. In contrast, co-expression of R1α with mutant L206R PKA_{cat} produced a unimodal distribution (**Figure 22H**), suggesting that even though the L206R mutation weakened PKA_{cat} association with AKAR4, disruption to R1α binding was much more severe, resulting in $K_I/K_e > 1$ and an inability to restrain PKA_{cat} activity. In contrast, co-expression of PKI with mutant L206R PKA_{cat}, recapitulates a bimodal distribution (**Figure 21D**), showing that PKI remains capable of blocking L206R PKA_{cat} activity. This is consistent with the earlier findings of weakened yet persistent binding between mutant PKA_{cat} and PKI (**Figure 20D**). Thus, these functional experiments reveal that the mutation L206R in the critical P+1 loop of PKA_{cat} is consequential not only with regards to its interactions with its regulators and inhibitors, but also its substrates.

Discussion

This study reports the use of the flow cytometer for quantitative live-cell FRET, demonstrating its utility both in providing a high-throughput assay for measuring protein-protein binding affinities, and also in revealing biological mechanisms through population-based analyses with FRET-based sensors. We show a simple procedure for absolute calibration of a commercially-available flow cytometer, using it to construct live-cell FRET binding curves that yield binding affinities comparable to those from ITC. With flow cytometric FRET, we demonstrate that a mutation in PKA_{cat} differentially

affects its interaction with its binding partners, resulting in unregulated but slowed catalysis of certain substrates. These results demonstrate that flow cytometric FRET represents a simple, fast and powerful approach to interrogating protein binding and function, providing an important adjunct to traditional biochemical tools. The increasing affordability, accessibility and capabilities of flow cytometry in recent years raise expectations that this methodology will find widespread use and development for years to come.

Strengths and Limitations of Flow Cytometric FRET

The major advantages of this strategy over *in vitro* approaches stem from the use of living mammalian cells as biochemical reactors. Proteins expressed in living mammalian cells receive the support of chaperones and accessory proteins, maturing with appropriate post-translational modifications, and thus tend to remain more stable within their native cellular contexts than without. With lengthy and costly protein purifications rendered unnecessary, binding assays can be accomplished with greater ease and speed, making larger scale studies of protein-protein interactions under various pharmacologic or genetic manipulations more practical. In addition to protein binding, flow cytometric FRET also permits population-level analyses of biological function in living cells via the ever-growing list of FRET-based biosensors. All in all, it is the strict accounting of every fluorescent species in addition to the FRET efficiency within each cell afforded by this approach that enables the engagement of biological problems from a quantitative and model-based perspective. In this light, our work here represents a few of many possible applications of flow cytometric FRET, and we anticipate that uses beyond those

suggested here will continue to surface, especially when coupled with cytometers with sorting capabilities, or automated flow cytometric platforms.

Assaying binding with flow cytometric FRET is not without limitations. These include the necessity of altering proteins with fluorescent protein tags, and the interference by unaccounted endogenous proteins. In many cases, the former problem is surmountable as protein tags can be designed such that they do not affect native protein function, in accordance with the modular nature of many proteins. The unwanted contribution of endogenous proteins, on the other hand, typically have their highest impact on high affinity interactions, as their effects can usually be minimized by the overexpression of fluorescently-tagged recombinant proteins. If necessary, genetic strategies for knocking down endogenous protein levels can also be employed. Lastly, flow cytometry is a high-throughput tool that requires the detachment of cells from their substrates for spectroscopic interrogation. As a result, not only are unique cell features such as cell morphology and local density lost, but select aspects of cell biology are also potentially altered by the resuspension of cells in solution. These issues likely only significantly affect a minority of proteins, and can be assessed by combining whole cell measurements of FRET with imaging via microscopy, or with more advanced automated imaging platforms [111].

Implications of L206R for PKA function

We determined that the Cushing's disease-causing L206R mutation within the P+1 loop of PKA_{cat} completely disrupts its interaction with PKA_{reg}, while largely preserving its binding to PKI. The differential effect on binding of L206R is consistent with prior

studies showing the ability of PKI but not PKA_{reg} to suppress the activity of L206R PKA_{cat} [102, 112]. Additionally, monitoring PKA activity using the FRET-based PKA sensor AKAR4 further reveals that this mutant PKA_{cat} has not only weakened binding to AKAR4, but also reduced catalytic activity. These findings are not unexpected, given that mutagenesis of other residues within the P+1 loop have been shown to interfere with catalytic activity [110]. However, they do conflict with reports that showed equal [113] or elevated [112] catalysis by L206R PKA_{cat} with the synthetic substrate Kemptide. The reason for this discrepancy is not clear, but may be due to the differential effects of the L206R mutation on different PKA substrates. Finally, we described a model of AKAR4 phosphorylation incorporating both substrate and inhibitor that allowed us to assess the relative "affinities" for substrate vs inhibitor (K_I/K_e) through the distribution of FRET efficiencies. These experiments independently confirmed the disparate effects on binding of PKA_{cat} L206R, and also illustrated the importance of an intermediary K_e for optimal switching of PKA activity. Overall, these findings hold key implications for understanding the pathogenesis of PKA_{cat} L206R mediated disease, and guiding future therapeutic strategies.

Figures and Tables

Primer	Sequence (5' to 3')
P01	atatggatccGTGAGCAAGGGCGAGGAG
P02	atatttaCTTGTACAGCTCGTCCATGC
P03	atatgcggccgctagatctCCCGTACAACTCCAGGCAGCAGGGCGGGTGCGTTGGGCCAG
P04	atattctagaagcttaTCGCATCATGGGGGCTACATAG
P05	gatctcgagctCCTGCTCCATCAATAGACAGAAGCACAAAACCTCCCCTGtaagcttata
P06	tataagcttaCAGGGGAGGTTTTGTGCTTCTGTCTATTGATGGAGCAGGagctcgagatc
P07	gatctcgagctCCTGCTCCATCAATAGAgAGAAGCACAAAACCTCCCCTGtaagcttata
P08	tataagcttaCAGGGGAGGTTTTGTGCTTCTcTCTATTGATGGAGCAGGagctcgagatc
P09	gatctcgagctCCTGCTCCATCAATAaagAGAAGCACAAAACCTCCCCTGtaagcttata
P10	tataagcttaCAGGGGAGGTTTTGTGCTTCTcttTATTGATGGAGCAGGagctcgagatc
P11	gatctcgagctCCTGCTCCATCAATAGACAGAAGCACAAgACCTCCCCTGtaagcttata
P12	tataagcttaCAGGGGAGGTcTTGTGCTTCTGTCTATTGATGGAGCAGGagctcgagatc
P13	gatctcgagctCCTGATATCCCACCACCTCGGCCACCAAAGCCACATCCAtaagcttata
P14	tataagcttaTGGATGTGGCTTTGGTGGCCGAGGTGGTGGGATATCAGGagctcgagatc
P15	gatctcgagctGAACCACCCCCGGTGGATAGGAACCTCAAGCCAGACAGAtaagcttata
P16	tataagcttaTCTGTCTGGCTTGAGGTTCCTATCCACCGGGGGTGGTTCagctcgagatc
P17	gatctcgagctAGGAACCTCAAGCCAGACAGAAAAGTCAAGCCGGCACCTtaagcttata
P18	tataagcttaAGGTGCCGGCTTGACTTTTCTGTCTGGCTTGAGGTTCCTagctcgagatc
P19	gatctcgagctTCCAACACTCCACCTCCaCGCCCaCCTAAGCCAAGCCATtaagcttata
P20	tataagcttaATGGCTTGGCTTAGGtGGGCGtGGAGGTGGAGTGTTGGAagctcgagatc
P21	gatctcgagctGAACCTCCCCCAGTGAATAGAGATCTCAAGCCTCAGAGGtaagcttata
P22	tataagcttaCCTCTGAGGCTTGAGATCTCTATTCACTGGGGGAGGTTCagctcgagatc
P23	gatctcgagctAATGTTCCACAGATTGATCGTACTAAAAAACCAGCAGTCtaagcttata
P24	tataagcttaGACTGCTGGTTTTTTAGTACGATCAATCTGTGGAACATTagctcgagatc
P25	gatctcgagctAAGCCACCTGTGGTGGACAGGTCCTTGAAACCTGGAGCAtaagcttata
P26	tataagcttaTGCTCCAGGTTTCAAGGACCTGTCCACCACAGGTGGCTTagctcgagatc
P27	gatctcgagctAAAGCTCCCATGGTGAATAGATCAACCAAGCCAAATTCCtaagcttata
P28	tataagcttaGGAATTTGGCTTGGTTGATCTATTCACCATGGGAGCTTTagctcgagatc
P29	atatggatccGGtAACGCtGCaGCtGCCAAG
P30	tataagcttaAAACTCAGAAAACTCCTTGCCACAC
P31	CTGAGTACCgGGCCCCTGAG
P32	CTCAGGGGCCcGGTACTCAG
P33	atatggtaccATGGAGTCTGGCAGTACCGC
P34	atatggatccGACAGACAGTGACACAAAACTGTTGTAC
P35	atatggtaccATGAGCATCGAGATCCCGGC
P36	atatagatctTGCAGTGGGTTCAACAATATCCATG
P37	CTGCAGTCGACGGTACCATGAC
P38	atatggatccTGACTCGGACTTAGCAGCTTCTC
P39	atatggtaccatgGGtAACGCtGCaGCtGCCAAG
P40	atatctagactaAAACTCAGAAAACTCCTTGCCACAC

Table 1

Table 1. List of primers used

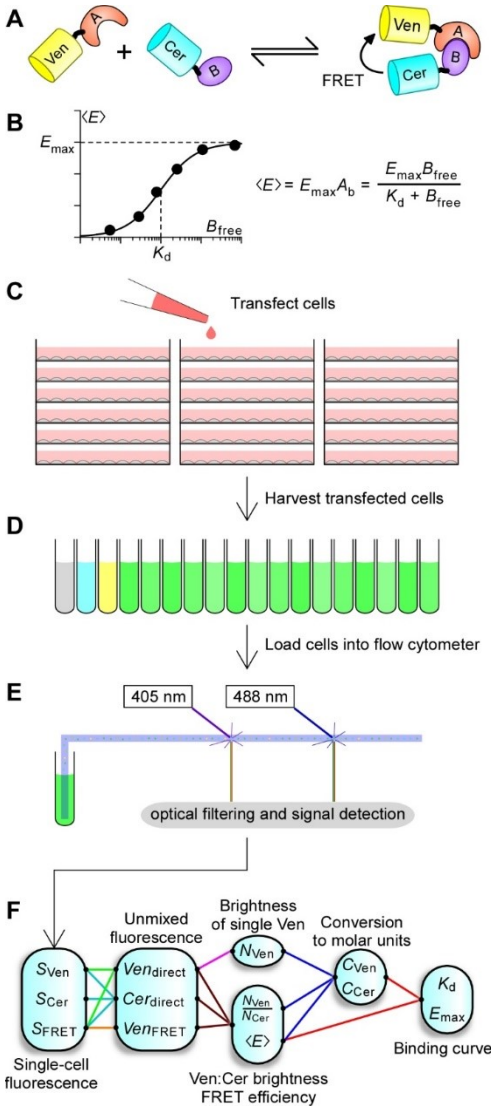

Figure 13

Figure 13: Flow cytometric FRET

A. Cartoon depicting binding partners *A* and *B* tagged with fluorescent proteins *Ven* (monomeric Venus) and *Cer* (monomeric Cerulean). **B.** FRET binding curves are obtained by plotting $\langle E \rangle$ as a function of either A_{free} or B_{free}. **C.—E.** HEK293(T) cells are cultured and transfected in 6 well plates, harvested into cytometer-compatible round bottom tubes, and loaded into the flow cytometer. **F.** Fluorescence signals are analyzed offline using custom software to yield the concentration of *Cer*, *Ven*, and $\langle E \rangle$, from which binding curves can be constructed.

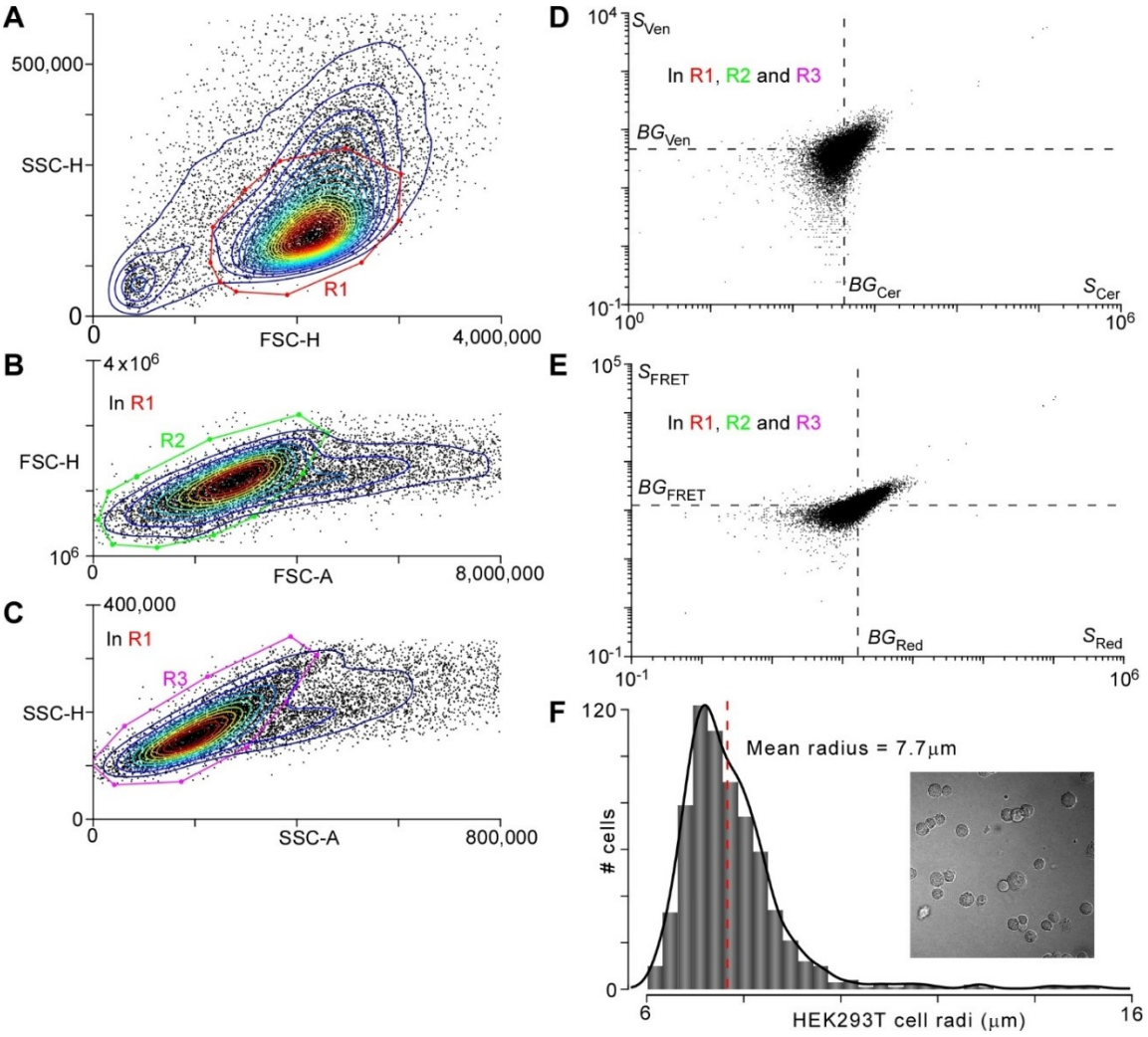

Figure 14

Figure 14. Selection, background fluorescence, and size of single cells.
A-C. Conventional flow cytometric strategies for excluding debris and non-viable, clumped HEK293T cells are employed. R1 gate in the FSC-H vs SSC-H plot selects viable cells, while R2 and R3 in the Height *vs* Area plots select for singlets. **D-E.** Background fluorescence levels are obtained from cells incubated in PEI without the addition of any plasmids, and calculated by taking the mean fluorescence of each channel, excluding 1% of outliers. **F.** Mean cell volume was approximated by measuring the mean radius of HEK293T cells and using the equation: $V = {}^{4}/_{3}\pi r^{3}$. An aliquot of cells prepared for flow cytometry was plated on a glass coverslip, and DIC images taken through a 40x objective (inset). Radii of cells were measured on Matlab, with distances calibrated using a micrometer.

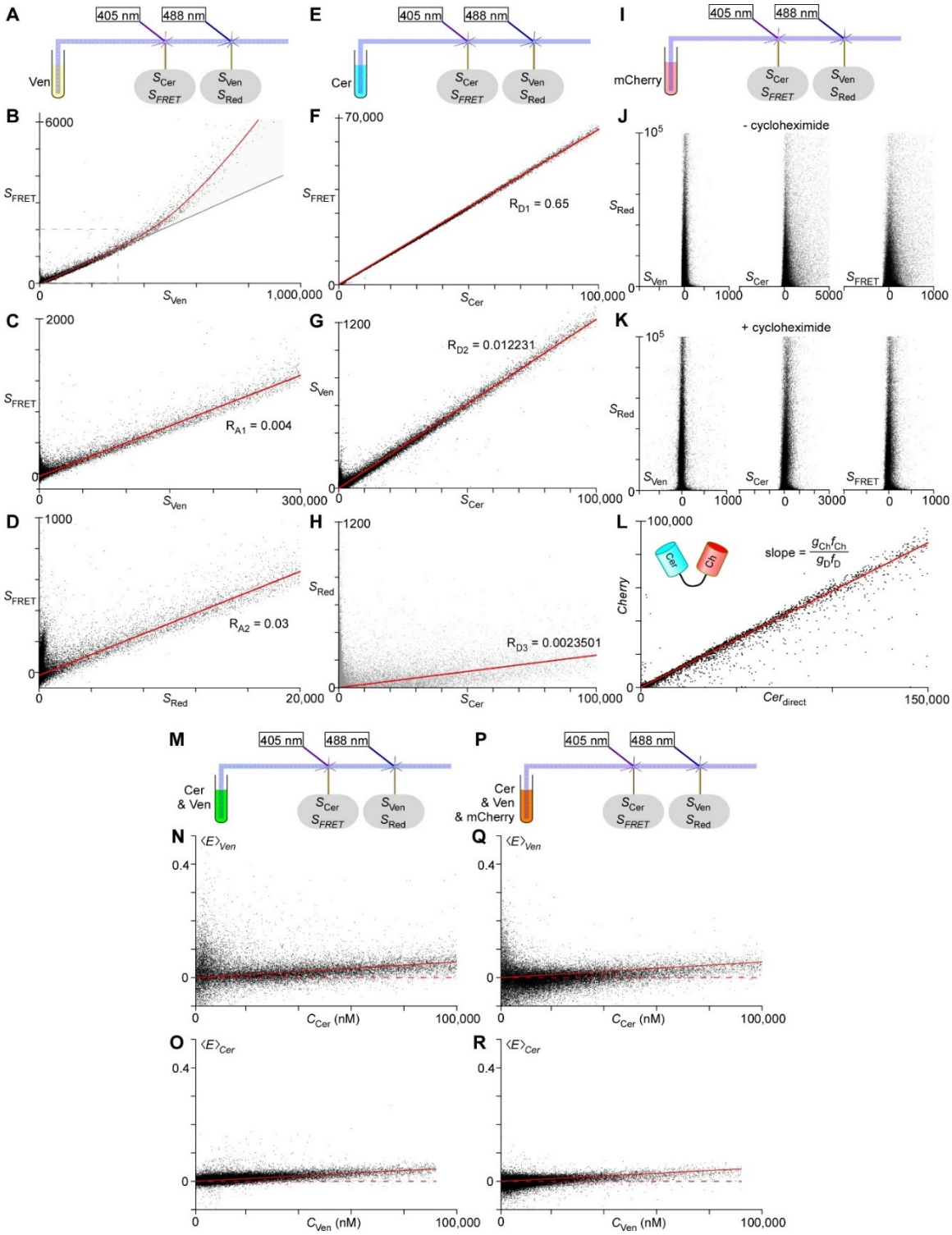

Figure 15

Figure 15. Accounting for cross-talk across channels.

A-D. Ven expressing cells are analyzed with the flow cytometer. **B.** Plotting S_{FRET} *vs* S_{Ven} shows that bleedthrough into the S_{FRET} channel becomes non-linear for very bright cells (S_{Ven} > 300,000). Below this threshold, as shown in **C** and subsequent panels, the relationship between channels is linear, allowing for linear decomposition of fluorescences from multiple fluorophores into their fundamental signals. **E-H.** Cer expressing cells are analyzed with the flow cytometer, yielding the constants R_{D1}, R_{D2} and R_{D3}. **I-L.** mCherry expressing cells are analyzed with the flow cytometer. **J.** Even though Ven and Cer fluorescences bleed into the S_{Red} channel (e.g. in **D** and **H**), mCherry fluorescence does not crosstalk with the other 3 channels in a correlated fashion. There is however, uncorrelated blue fluorescence into the S_{Cer} and S_{FRET} channels, due to mCherry transitioning through a "blue state" as it matures. **K.** With the pulsed addition of cycloheximide, mCherry is allowed to mature resulting in eradication of the blue fluorescence. **L.** Cells expressing Cer-22aa-mCherry are analyzed with the flow cytometer. Since the fluorophores are guaranteed to be expressed in a 1:1 ratio, the slope of the Cer_{direct} *vs Cherry* curve allows us to determine the brightness of a single mCherry molecule as measured by our flow cytometer. **M-O.** Cells expressing both Cer and Ven exhibit a linear, concentration-dependent increase in FRET. This is due to FRET between molecules of Cer and Ven that interact transiently during a random collisional event. Since the chance of collisions increase linearly with concentration, so too does the FRET. **P-R.** Similarly, collisional FRET is seen in cells expressing Cer, Ven and mCherry. Notably, the addition of mCherry does not affect the quantitation of FRET between Cer and Ven.

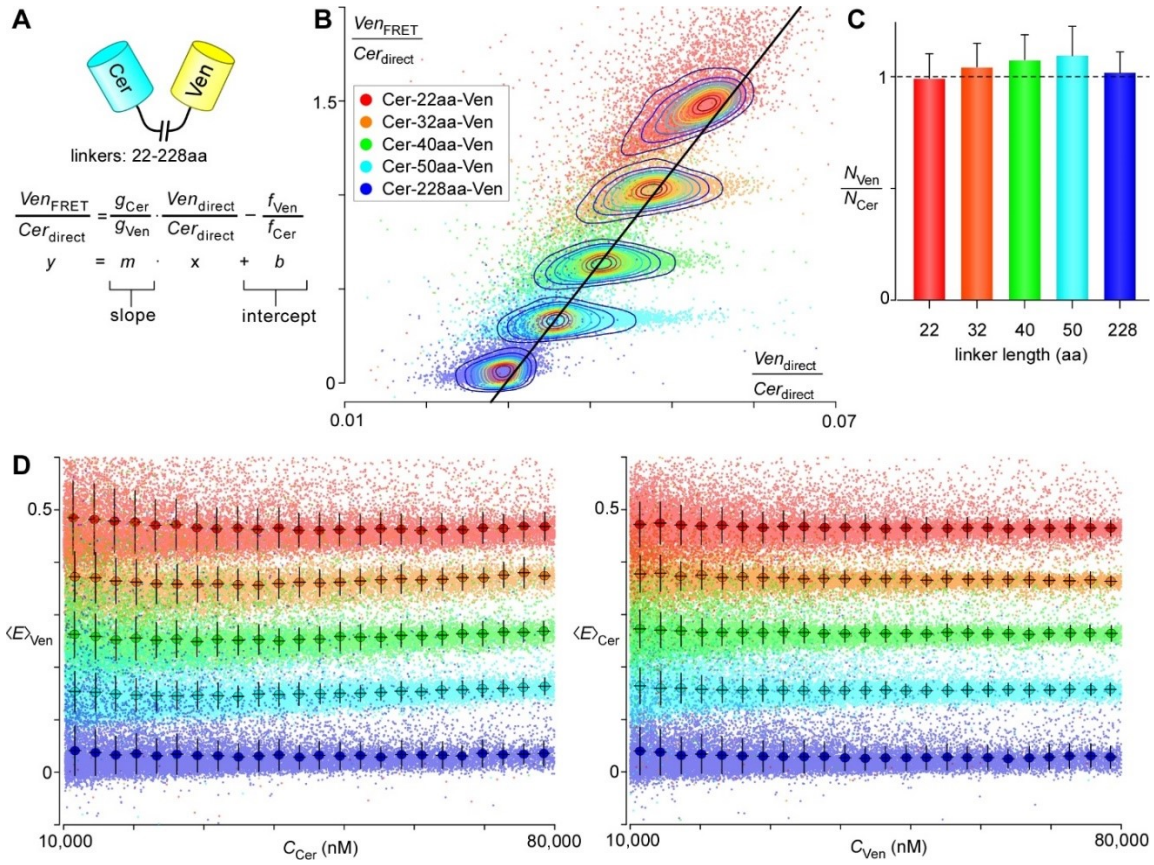

Figure 16

Figure 16: Calibrating the flow cytometer for FRET

A. The ratios g_{Cer}/g_{Ven} and f_{Ven}/f_{Cer} can be discerned from Cer-Ven dimers of varying linker lengths using the following linear equation. **B.** Each cell expressing a particular Cer-Ven dimer is plotted as a single colored dot on the graph. Contour lines are overlaid on top of each Cer-Ven dimer to show the areas of highest density. **C.** The mean ± SD of N_{Ven}/N_{Cer} for each dimer is plotted. **D.** Plots of $\langle E \rangle_{Ven}$ and $\langle E \rangle_{Cer}$ for the five Cer-Ven dimers, with each point representing data from a single cell.

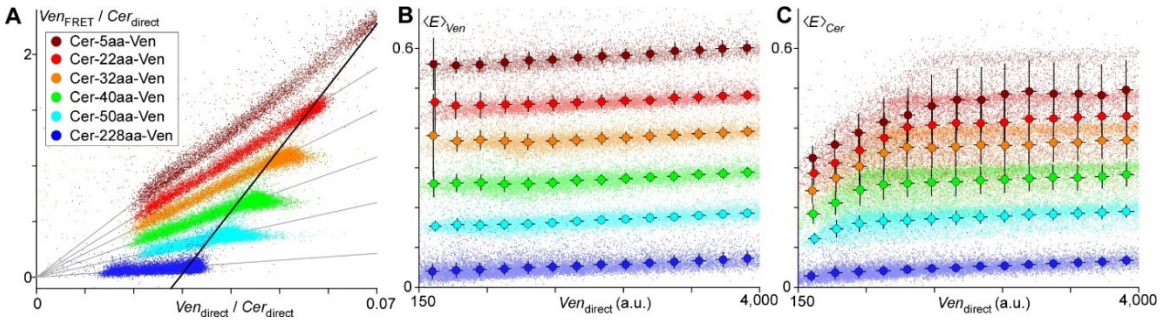

Figure 17

Figure 17. Effects of immature FPs on FRET quantitation.
A. Plots of Ven_{FRET}/Cer_{direct} *vs*. $Ven_{direct}/Cer_{direct}$ in cells expressing Cer-Ven dimers yielded a smearing of points following a line towards the origin. **B.** FRET efficiency from Ven's perspective increases with decreasing linker length, with small variance cell-to-cell and an appropriate linear, concentration-dependent increase in FRET. **C.** In contrast, FRET efficiency from Cer's perspective has large variance from cell-to-cell, with a significant proportion of cells having lower FRET than expected, especially for dim cells expressing Cer-Ven dimers with smaller linker lengths.

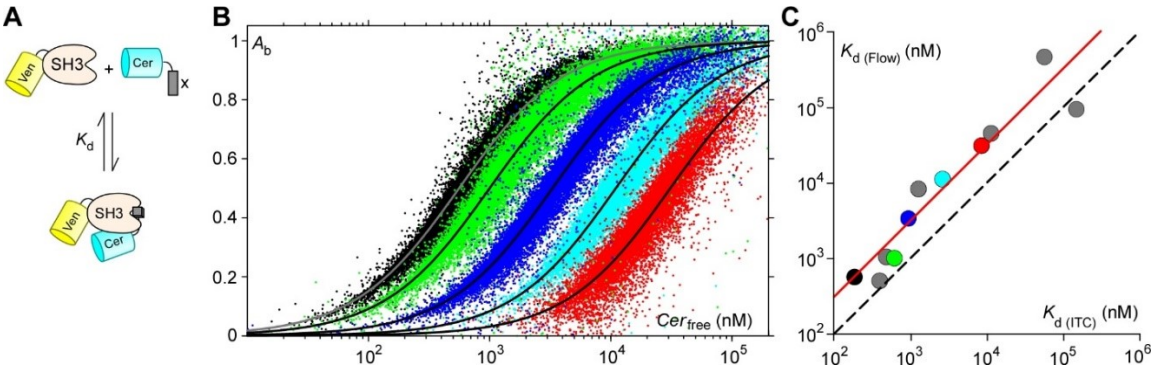

Figure 18

Figure 18: Flow cytometric FRET binding curves

A. Diagram of the binding reaction between Ven-SH3 (Mona/Gads) and Cer-X, where X represents one of several binding partners of Mona/Gads. **B.** FRET binding curves for Ven-SH3 in complex with SLP-76, aa.231-243 (black), Gab1, aa.515-527 (green), USP8, aa.403-415 (blue), SLP-76 (D236K), aa.231-243 (cyan), and SLP-76 (K240R), aa.231-243 (red). Their respective binding affinities were K_d = 0.8, 1, 4, 7 and 24 μM, with E_{max} = 0.39, 0.4, 0.37, 0.35 and 0.35 respectively. **C.** Comparison between dissociation constants measured by flow cytometry (y-axis) and those measured by ITC (x-axis). On average, affinities measured by flow cytometry are ~3-fold less.

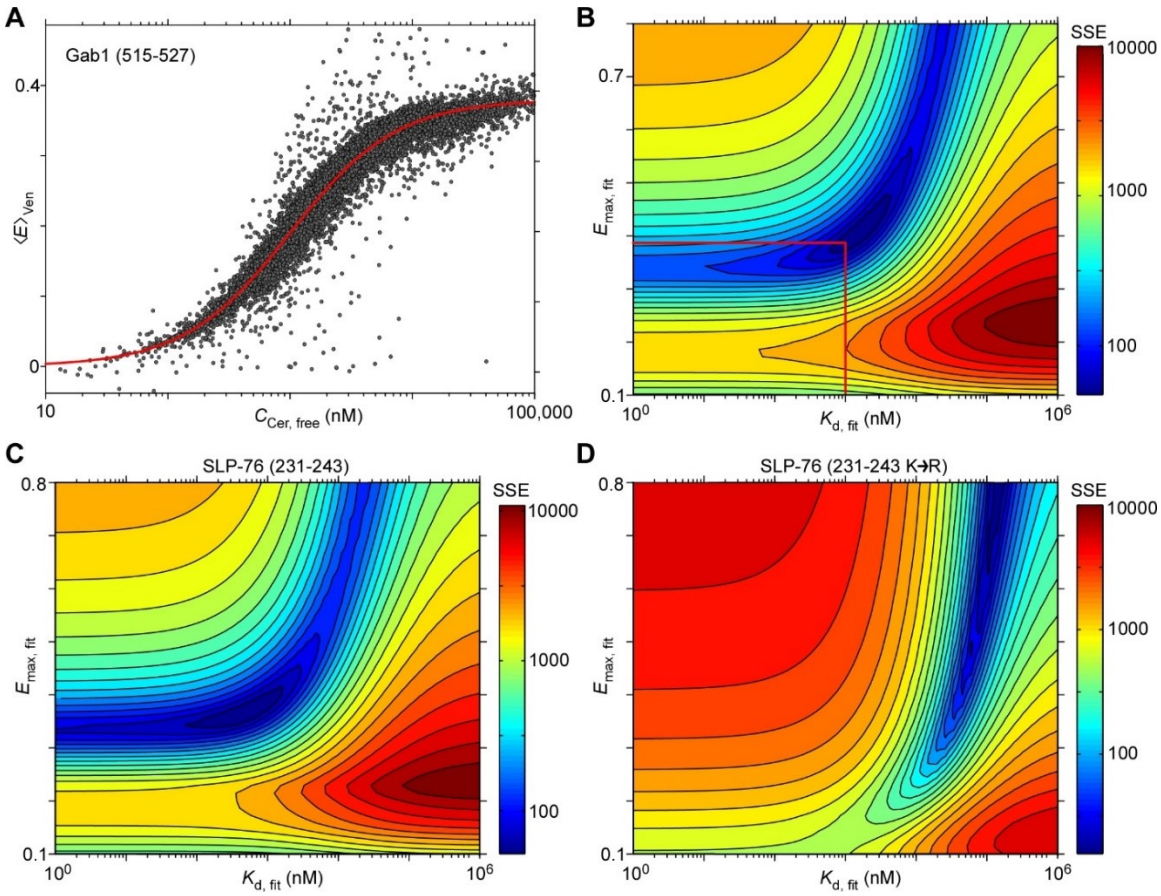

Figure 19

Figure 19. Determining K_d and E_{max}.
A. Binding curve of Cer-Gab1(515-527) *vs* Ven-Gads(SH3) with K_d = 1000 nM and E_{max} = 0.38. **B.** Sum of squared errors (SSE) landscape of binding partners in **A**. An error minimum is clearly identified. **C.** SSE landscape for Cer-SLP76 (231-243) *vs* Ven-Gads(SH3), a pair with higher binding affinity. The error minimum becomes shallower for high affinity binders (extended blue contours horizontally to the left). **D.** SSE landscape for Cer-SLP76 (231-243 K→R), a pair with low binding affinity. Because of the inability to obtain data points near E_{max}, the landscape does not demonstrate a true minimum. Nonetheless, with educated constraints on E_{max}, the K_d can still be reasonably estimated.

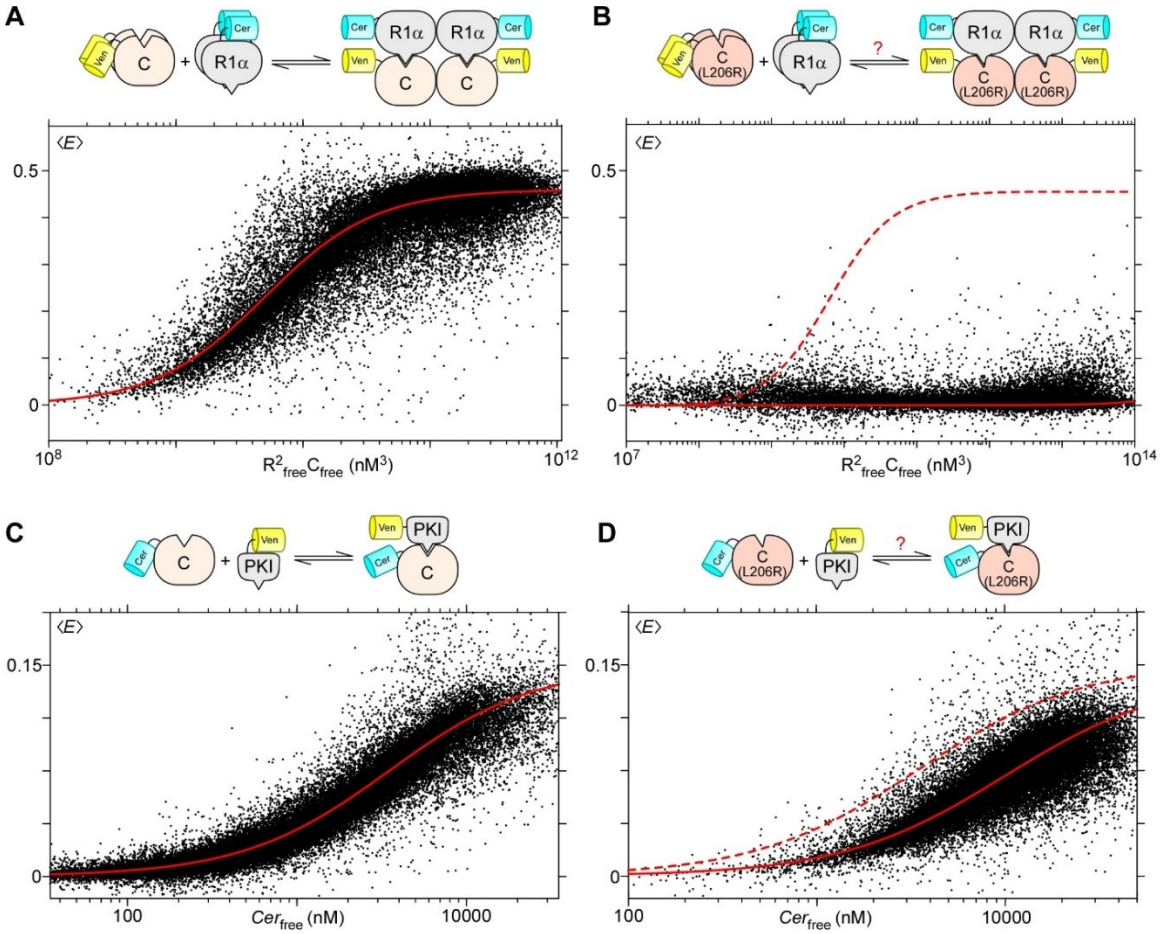

Figure 20

Figure 20: Differential effects on binding by PKAcat L206R

A. Fluorescent-protein tagged PKA_{cat} and PKA_{reg} fall on a FRET binding curve, with E_{max} = 0.46 and $K_d \approx 1.7$ µM. **B.** Binding curves between PKA_{cat} (L206R) and PKA_{reg} show complete lack of binding. **C.** and **D.** Similarly, these scatter plots show binding curves between PKI and either PKA_{cat} or PKA_{cat} L206R, both of which bind with E_{max} = 0.16, and K_d = 3.5 and 10 µM respectively.

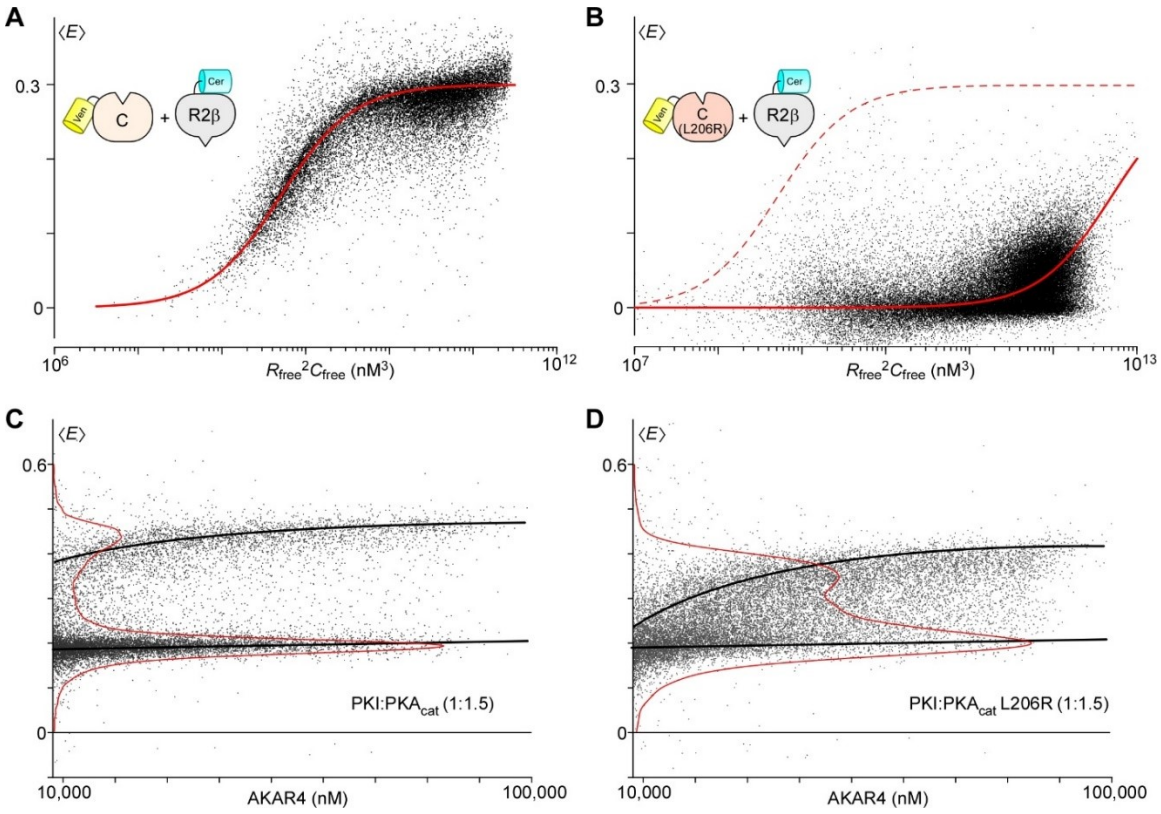

Figure 21

Figure 21. Further structure-function studies of PKA.
A. FRET curves demonstrating binding between Ven-PKA_{cat} and R2β-Cer. **B.** Similarly, cells expressing Ven-PKA_{cat} (L206R) with R2β-Cer show lack of binding. **C.** AKAR4 in the presence of coexpressed PKI and PKA_{cat} show a bimodal activity level. **D.** AKAR4 in the presence of coexpressed PKI and PKA_{cat} (L206R) again show a blunted bimodal distribution of AKAR4 phosphorylation.

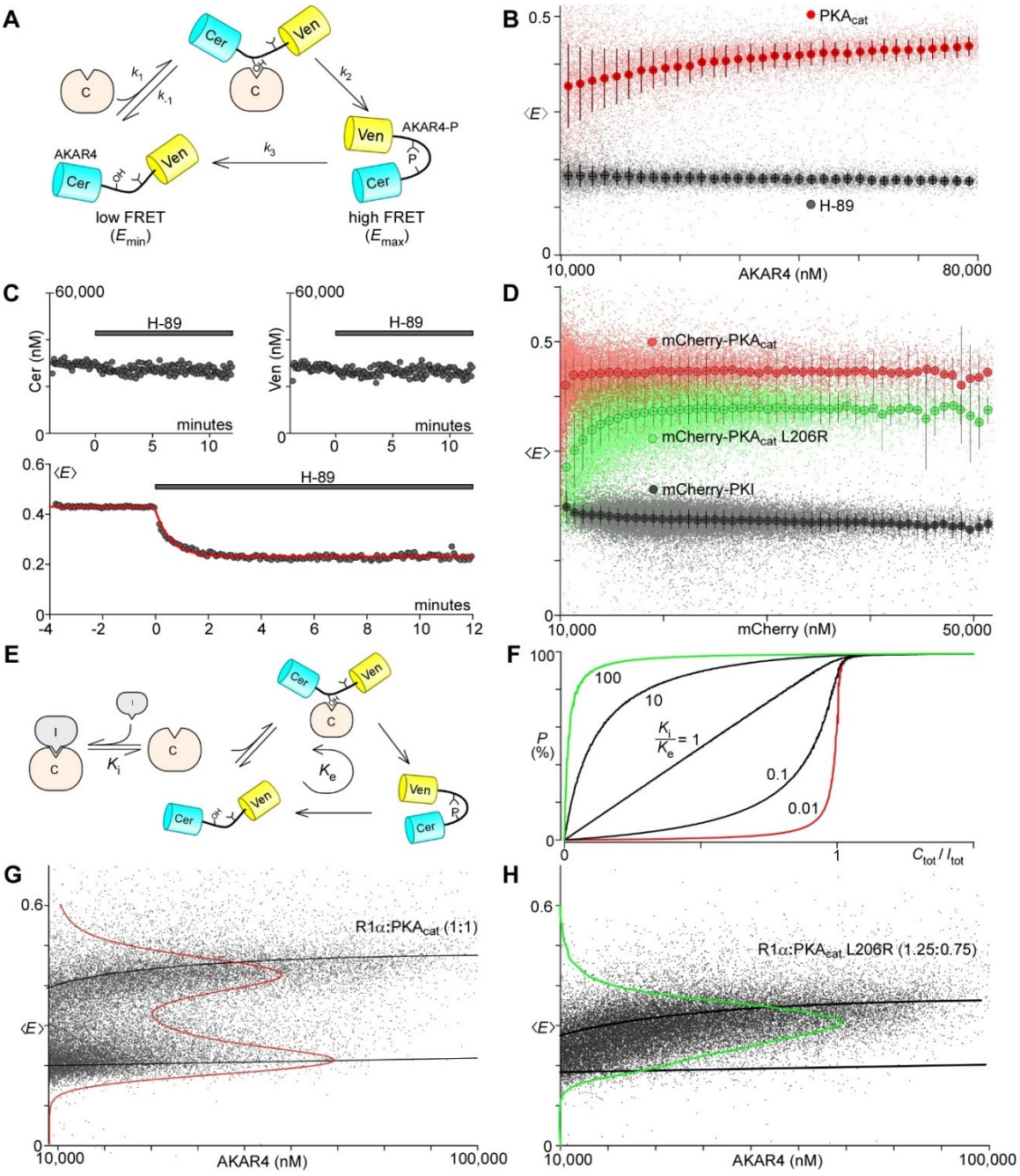

Figure 22

Figure 22: Enzyme kinetics of PKA L206R revealed through FRET-based PKA sensor.

A. Diagram illustrating the different states of the PKA activity sensor AKAR4. In its unphosphorylated state, it has low FRET (E_{min}). Phosphorylation through a Michaelis-Menten scheme (k_{-1}, k_1, k_2) results in a state of high FRET (E_{max}). Dephosphorylation (k_3)

occurs through endogenous phosphatases. **B.** Scatter plot of single cell FRET efficiency as a function of AKAR4 expression, either with PKA_{cat} overexpressed (red) or with H-89 (100 μM) in the solution (black). **C.** H-89 (100 μM) was added to the solution containing cells expressing AKAR4 and PKA_{cat} at time zero. While measurements of Cer and Ven concentrations were unaffected by addition of H-89 (top), the average $\langle E \rangle$ declined exponentially with τ = 33 sec (red curve), reflecting dephosphorylation of AKAR4 driven by endogenous phosphatases. **D.** Scatter plot of single cell FRET efficiency *vs* concentration of mCherry-tagged proteins. While expression of PKI (black) caused a decrease towards E_{min}, expression of PKA_{cat} (red) resulted in a dramatic increase in $\langle E \rangle$. On the other hand, expression of PKA_{cat} L206R resulted in a gentler ascent in $\langle E \rangle$, and settling a lower plateau. **E.** Scheme outlining the addition of an inhibitor. K_I represents the dissociation constant of the inhibitor, whereas K_e is like an "aggregate dissociation constant" for the AKAR4 system. **F.** Solution to the model diagrammed in **e.** where P (percent of AKAR4 phosphorylated) is plotted versus C_{tot}/I_{tot} (PKA_{cat} to inhibitor ratio) for different values of K_I/K_e. **G.** and **H.** Single cell FRET efficiencies of cells transfected with PKA_{reg} together with either PKA_{cat} or PKA_{cat} L206R. Mean results of either PKA_{cat} (L206R in **H.**) or PKA_{reg} alone are represented by thick black lines, while population FRET distributions are overlaid in red.

Appendix

IA. Simulation code for 2 compartment model

```
%% full model with CDF.
clear;
% voltage parameters
tstart = 0;
tend = 250;
holdingV = -70;
stepV = -20;
tpulse = 1;
stimtimes = [50 60 70 80 90 100 110 120 130 140 150 160 170 180 190];
startind = 1;
tdelay = stimtimes(startind);
sampkHz = 100;
APWaveform(:,1) = 0:1/sampkHz:tend';
APWaveform(:,2) = APWaveform(:,1)*0 + holdingV;

for k=1:length(stimtimes)
    ind = (stimtimes(k))*sampkHz:(stimtimes(k) + tpulse)*sampkHz;
    if max(ind)<size(APWaveform,1)
        APWaveform(ind,2) = stepV;
    end
end

% synapse params
Nchan = 60000;
Rsphere = 8; % um
Rcyto = 7.9;
restingCa = 0.04; % uM
k12 = 0.6e-12;
k21 = k12;
ke =  0.6e-13;

% channel params (see channel2cptCDFmodel.m)
A = 3.3;
RTf = 26;
z1 = 1.1;
B = 1.1;
z2 = 0.9;
k5 = 3.4;
f = 2;
k5_f = k5*f;
m=1;
n=1.1;
Vh3 = -52; E = 3; F = 2.6; H = 15;
k_5p = [Vh3, E, F, H];
k_5_fp = 0.6154; % g = 1/k_5_fp;

ka = 0.1; k_a = 0.13;
kb = 0.44; k_b = 0.19;
nHill = 1.8;
Cai = restingCa; Cao = 2000; RTzF = 13; scaleF = 5e-6;  % gives ~0.13pA at 0 mV

% initial values
yinitial = zeros(20,1);
yinitial(1) = 1;  %channel starts at C1
yinitial(19) = restingCa;  % resting Ca in shell
yinitial(20) = restingCa;  % resting Ca in cyto
options = odeset('RelTol',1e-3,'OutputFcn',@odetpbar,'MaxStep',1);

% simulate
[T,Y] = ode23s(@(t,y) channel2cptCDFmodel(t,y,APWaveform,A,B,z1,z2,k5,k_5p,k5_f,k_5_fp,...
    ka,k_a,kb,k_b,nHill,m,n,k12,k21,ke,Rsphere,Rcyto,Nchan,Cai,Cao,RTzF,scaleF),...
    [tstart tend],yinitial,options);

PO = Y(:,6)+Y(:,12)+Y(:,18);
V = APWaveform(:,2);
i = interp1(APWaveform(:,1),...
    scaleF*V.*(Cai - Cao*exp(-V/RTzF))./(1 - exp(-V/RTzF)), T); %GHK
Isim = (Nchan*PO).*i; % in pA
Ca1  = Y(:,19);
Ca2 =  Y(:,20);

%% plots
figure; set(gcf,'color','w');
hV = subplot(3,2,1);
plot(T,interp1(APWaveform(:,1),APWaveform(:,2),T));
title('mV vs t');

hCas = subplot(3,2,2);
plot(T,Ca1); set(gca,'YScale','lin');
title('Ca local vs t');

hCag = subplot(3,2,4);
plot(T,Ca2); set(gca,'YScale','lin');
```

```
title('Ca global vs t');

hCaB = subplot(3,2,6);
[maxI,minI] = peakdet(Isim,-min(Isim)/2);
line(T(minI(:,1)),minI(:,2),'marker','.','linestyle','none','MarkerSize',20);
ystart = minI(1,2);
yend = ystart*1.83;
tau = 30;
yexp = yend + (ystart-yend)*exp(-(T-tdelay)/tau);
yexp(find(T<tdelay)) = ystart;
line(T,yexp,'color','r');
set(gca,'YScale','lin');
title('Ipeak vs t');

hPO = subplot(3,2,3);
plot(T,PO);
% line(get(gca,'XLim'),[kb/(kb+k_b) kb/(kb+k_b)],'color','g'); % POmax
title('Po vs t');

hI = subplot(3,2,5);
plot(T,Isim);
title('pA vs t');

linkaxes([hV,hPO,hI],'x');
linkaxes([hCas,hCag,hCaB],'x');
set(hV,'XLim',[tstart tend]);
set(hCas,'XLim',[tstart tend]);

function dy =
channel2cptCDFmodel(t,y,APWaveform,A,B,z1,z2,k5,k_5p,k5_f,k_5_fp,ka,k_a,kb,k_b,nHill,m,n,k12,k21,ke,Rsphere,Rcyto,Nchan
,Cai,Cao,RTzF,scaleF)
%       k1          k2          k3          k4          k5
% C1  <--->   C2  <--->   C3  <--->   C4   <--->   C5   <--->   O6
%       k_1         k_2         k_3         k_4         k_5
% ||  kaCa2   ||      k_a ||          ||          ||
%       k1          k2          k3          k4          k5
% C7  <--->   C8  <--->   C9  <--->   C10  <--->   C11  <--->   O12
%       k_1         k_2         k_3         k_4         k_5
% ||  kb      ||      k_b ||          ||          ||
%       k1          k2          k3          k4          k5_f
% C13  <--->  C14 <--->   C15 <--->   C16  <--->   C17  <--->   O18
%       k_1         k_2         k_3         k_4          k_5_f

% let k1 = 4*kf, k2 = 3*kf, k3 = 2*kf, k4 = kf
% kf = A*exp(V*z1/RTf)
% and k_1 = kr, k_2 = 2*kr, k_3 = 3*kr, k_4 = 4*kr
% kr = B*exp(-(V*z2)/RTf).
% k5 and k5_f constant
% k_5 voltage dependent, k_5 = H/(1+exp((V-Vh3)/E)) + F;
% k_5_f = k_5/g
% cdf represents enhanced opening, with f>1, otherwise f=1.


%   Ica           k12           ke
% ---> Cashell  <--->   Cacyto  <--->  restingca
%                 k21           ke


RTf = 26;
V = interp1(APWaveform(:,1),APWaveform(:,2),t);

kf = A*exp(V*z1/RTf);
kr = B*exp(-(V*z2)/RTf);
k1 = 4*kf; k2 = 3*kf; k3 = 2*kf;  k4 = kf;
k_1 = kr; k_2 = 2*kr; k_3= 3*kr;  k_4 = 4*kr;

Vh3 = k_5p(1); E = k_5p(2); F = k_5p(3); H = k_5p(4);
k_5 = H/(1+exp((V-Vh3)/E)) + F;
k_5_f = k_5*k_5_fp;

f = k5_f/k5;
g = 1/k_5_fp;

% volume of calyx ~0.5pL
% 100 um^3 = 0.1pL
Vcyto = 4/3*pi*Rcyto^3*1e-15; % in L
Vshell = 4/3*pi*Rsphere^3*1e-15 - Vcyto; % in L
surfaceA = 4*pi*(Rcyto*1e-6)^2;

dy = zeros(20,1);
% diff equations
dy(1) =       -(m*k1+ka*y(19)^nHill)*y(1)             +  m*k_1*y(2) + k_a*y(7);
dy(2) = -(m*k_1+m*k2+ka*y(19)^nHill)*y(2) +  m*k1*y(1) +  m*k_2*y(3) + k_a*y(8);
dy(3) = -(m*k_2+m*k3+ka*y(19)^nHill)*y(3) +  m*k2*y(2) +  m*k_3*y(4) + k_a*y(9);
dy(4) = -(m*k_3+m*k4+ka*y(19)^nHill)*y(4) +  m*k3*y(3) +  m*k_4*y(5) + k_a*y(10);
dy(5) = -(m*k_4  +k5+ka*y(19)^nHill)*y(5) +  m*k4*y(4) +    k_5*y(6) + k_a*y(11);
dy(6) =   -(k_5     +ka*y(19)^nHill)*y(6) +    k5*y(5)               + k_a*y(12);

dy(7) =      -(m*k1      +k_a+kb)*y(7)              +  m*k_1*y(8) + k_b*y(13)  +  ka*y(19)^nHill*y(1);
dy(8) =      -(m*k2+m*k_1+k_a+kb)*y(8) +  m*k1*y(7) +  m*k_2*y(9) + k_b*y(14)  +  ka*y(19)^nHill*y(2);
dy(9) =      -(m*k3+m*k_2+k_a+kb)*y(9) +  m*k2*y(8) +  m*k_3*y(10)+ k_b*y(15)  +  ka*y(19)^nHill*y(3);
```

```
dy(10) =     -(m*k4+m*k_3+k_a+kb)*y(10)+  m*k3*y(9) +  m*k_4*y(11)+ k_b*y(16)  +  ka*y(19)^nHill*y(4);
dy(11) =     -(k5  +m*k_4+k_a+kb)*y(11)+  m*k4*y(10)+    k_5*y(12)+ k_b*y(17)  +  ka*y(19)^nHill*y(5);
dy(12) =            -(k_5+k_a+kb*f)*y(12) + k5*y(11)              + k_b/g*y(18)+  ka*y(19)^nHill*y(6);

dy(13) =              -(n*k1+k_b)*y(13)              +  n*k_1*y(14)       +           kb*y(7);
dy(14)=         -(n*k_1+n*k2+k_b)*y(14)+  n*k1*y(13) +  n*k_2*y(15)       +           kb*y(8);
dy(15)=         -(n*k_2+n*k3+k_b)*y(15)+  n*k2*y(14) +  n*k_3*y(16)       +           kb*y(9);
dy(16)=         -(n*k_3+n*k4+k_b)*y(16)+  n*k3*y(15) +  n*k_4*y(17)       +           kb*y(10);
dy(17)=         -(n*k_4+k5_f+k_b)*y(17)+  n*k4*y(16) +  k_5_f*y(18)       +           kb*y(11);
dy(18)=         -(k_5_f   +k_b/g)*y(18)+  k5_f*y(17)                      +         kb*f*y(12);

% Calculate Ca extrusion part
% calculate unitary current for given voltage.

i = scaleF*V*(Cai - Cao*exp(-V/RTzF))/(1 - exp(-V/RTzF));

% calculate total Ca2+ current
ICa = (Nchan*(y(6)+y(12)+y(18)))*i*1e-12*1e-3;  % in units of A (C/ms)

% convert to Ca
F = 9.64*1e4*1e-6; % C/umol
Caflux = -ICa/(2*F); %umol/ms

dy(19)= (Caflux + k21*y(20) -  k12*y(19))/Vshell;
dy(20)= (k12*y(19) - (k21 + ke)*y(20) + ke*Cai)/Vcyto;


function [maxtab, mintab]=peakdet(v, delta, x)
%PEAKDET Detect peaks in a vector
%        [MAXTAB, MINTAB] = PEAKDET(V, DELTA) finds the local
%        maxima and minima ("peaks") in the vector V.
%        MAXTAB and MINTAB consists of two columns. Column 1
%        contains indices in V, and column 2 the found values.
%
%        With [MAXTAB, MINTAB] = PEAKDET(V, DELTA, X) the indices
%        in MAXTAB and MINTAB are replaced with the corresponding
%        X-values.
%
%        A point is considered a maximum peak if it has the maximal
%        value, and was preceded (to the left) by a value lower by
%        DELTA.

% Eli Billauer, 3.4.05 (Explicitly not copyrighted).
% This function is released to the public domain; Any use is allowed.

maxtab = [];
mintab = [];

v = v(:); % Just in case this wasn't a proper vector

if nargin < 3
  x = (1:length(v))';
else
  x = x(:);
  if length(v)~= length(x)
    error('Input vectors v and x must have same length');
  end
end

if (length(delta(:)))>1
  error('Input argument DELTA must be a scalar');
end

if delta <= 0
  error('Input argument DELTA must be positive');
end

mn = Inf; mx = -Inf;
mnpos = NaN; mxpos = NaN;

lookformax = 1;

for i=1:length(v)
  this = v(i);
  if this > mx, mx = this; mxpos = x(i); end
  if this < mn, mn = this; mnpos = x(i); end

  if lookformax
    if this < mx-delta
      maxtab = [maxtab ; mxpos mx];
      mn = this; mnpos = x(i);
      lookformax = 0;
    end
  else
    if this > mn+delta
      mintab = [mintab ; mnpos mn];
      mx = this; mxpos = x(i);
      lookformax = 1;
    end
  end
end
```

IIA. Quantitative FRET with flow cytometry

For a cell expressing both Cer and Ven, the background-subtracted fluorescence signals *S* measured at each of the 3 channels is represented by the following equations:

$$S_{Cer} = N_{Cer} g_{Cer,C1} f_{Cer,C1} (1 - \langle E \rangle_{Cer}) \quad (1)$$

$$S_{Ven} = N_{Ven} g_{Ven,C2} f_{Ven,C2} + N_{Cer} g_{Cer,C2} f_{Cer,C2} (1 - \langle E \rangle_{Cer}) \quad (2)$$

$$S_{FRET} = N_{Cer} g_{Cer,C3} f_{Cer,C3} (1 - \langle E \rangle_{Cer}) + N_{Ven} g_{Ven,C3} f_{Ven,C3} + N_{Ven} g_{Cer,C3} f_{Ven,C3} \langle E \rangle_{Ven} \quad (3)$$

where *g* and *f* are as defined in the main text, being a function of fluorophore properties as well as the filter wavelengths used, and $\langle E \rangle_{Cer}$ is proportional to the fraction of Cer bound to Ven while $\langle E \rangle_{Ven}$ is proportional to the fraction of Ven bound to Cer. In general, $\langle E \rangle_{Cer} \neq \langle E \rangle_{Ven}$ except when Cer and Ven molecules interact in a 1:1 fashion. *C1*, *C2*, and *C3* are shorthand for the filters used for each channel respectively. The constants R_{D1}, R_{D2}, and R_{A1} obtained from single fluorophore experiments allow the simplification of equations (1)-(3) to:

$$Cer_{direct} = R_{D1} S_{Cer} = N_{Cer} g_{Cer} f_{Cer} (1 - \langle E \rangle_{Cer}) \quad (4)$$

$$Ven_{direct} = R_{A1} (S_{Ven} - R_{D2} S_{Cer}) = N_{Ven} g_{Ven} f_{Ven} \quad (5)$$

$$Ven_{FRET} = S_{FRET} - Cer_{direct} - Ven_{direct} = N_{Ven} g_{Cer} f_{Ven} \langle E \rangle_{Ven} \quad (6)$$

where the notation for *g* and *f* can now be simplified since they all refer to the filters in channel *C*3. Cer_{direct} and Ven_{direct} refer to fluorescences by Cer or Ven due to direct excitation by the 405nm laser, while Ven_{FRET} is the fluorescence of Ven due to "excitation" of Ven by Cer through FRET. These measurable quantities can now be manipulated to yield the desired FRET efficiencies:

$$\langle E \rangle_{Ven} = \frac{g_{Ven}}{g_{Cer}} \frac{Ven_{FRET}}{Ven_{direct}} \quad (7)$$

$$\langle E \rangle_{Cer} = \frac{Ven_{FRET}}{Ven_{FRET} + {f_{Ven}}/{f_{Cer}} \, Cer_{direct}} \quad (8)$$

where equation (6) was rewritten as $Ven_{FRET} = N_{Cer} g_{Cer} f_{Ven} \langle E \rangle_{Cer}$ to yield equation (8). The process for determining *g* and *f* are described in the main text.

IIB. Binding model for PKA_{reg} and PKA_{cat}

A simplified model used to describe the highly cooperative binding between the catalytic and regulatory subunits of PKA can be represented as:

$$2\,R + 2\,C \underset{K_d^3}{\rightleftharpoons} R_2C_2$$

The steady-state equations for this binding model are represented as equations (9)-(11), which can be combined to form a 4th-order polynomial to solve for R_2C_2:

$$R_2C_2 = K_d^3 \cdot R^2 \cdot C^2 \quad (9)$$
$$R = R_{tot} - 2R_2C_2 \quad (10)$$
$$C = C_{tot} - 2R_2C_2 \quad (11)$$

A binding curve is obtained by plotting the fraction of C bound against R^2C as shown in equation (12), enabling an estimation of the K_d of the reaction.

$$\frac{2R_2C_2}{C_{tot}} = \frac{2R^2C}{K_d^3 + 2R^2C} \quad (12)$$

IIC. AKAR4 activity with PKA_{cat}

The model depicted in **Figure 5A** is represented by the following rearranged steady-state equations, where A and A^* represent the unphosphorylated and phosphorylated AKAR4, C represents free PKA_{cat} and CA represents the PKA_{cat}-AKAR4 complex:

$$CA = \frac{k_3A^*}{k_2} \quad (13)$$

$$A = \frac{A^*k_3(k_{-1} + k_2)}{C \cdot k_1 \cdot k_2} \quad (14)$$

The fraction of AKAR4 phosphorylated (P), which is proportional to mean FRET efficiency as expressed by main-text equation (11), is written as:

$$P = \frac{A^*}{A^* + A + CA}$$
$$= \frac{A^*}{A^* + \frac{A^*k_3(k_{-1}+k_2)}{C \cdot k_1 \cdot k_2} + \frac{k_3A^*}{k_2}}$$

$$= \frac{k_2}{k_2+k_3}\frac{C}{\frac{(k_{-1}+k_2)k_3}{k_1(k_2+k_3)}+C} \quad (15)$$

Substituting $y_{max} = k_2/(k_2 + k_3)$, $K_e = K_m(1 - y_{max})$, and $K_m = (k_{-1} + k_2)/k_1$ in equation (15) yields equation (12) in the main text. Of note, we make the assumption that $C \sim C_{tot}$, which is a reasonable assumption given that K_m and k_2 are large.

IID. AKAR4 activity in presence of an inhibitor

Addition of a competitive inhibitor of PKA_{cat} (as with PKA_{reg} or PKI) results in the scheme in **Figure 5E**. At steady-state, the fraction of phosphorylated AKAR4 (*P*) remains as written in equation (15), except the amount of free PKA_{cat} is now a function of C_{tot}, I_{tot} and K_I. With *I* representing free inhibitor and *CI* now representing the PKA_{cat}-inhibitor complex, we have the following equations:

$$C_{tot} = C + CA + CI \quad (16)$$

$$A_{tot} = A + A^* + CA \quad (17)$$

$$A^* = \frac{k_2}{k_3}(CA) \quad (18)$$

$$CI = \frac{C \cdot I_{tot}}{C + K_I} \quad (19)$$

$$A = (CA)\frac{k_2 + k_{-1}}{Ck_1} \quad (20)$$

Combining equations (16)-(20), we obtain a quadratic equation with the solution:

$$C = \frac{(C_{tot} - I_{tot} - K_I) + \sqrt{(K_I - C_{tot} + I_{tot})^2 + 4K_I C_{tot}}}{2} \quad (21)$$

Inserting equation (21) into equation (15) yields the curves seen in **Figure 5F**. However, what factors determine whether the *P vs* C_{tot}/I_{tot} curve is convex or concave? We can rewrite equation (21) in terms of the new variables $r = C_{tot}/I_{tot}$ and $i = K_I/I_{tot}$:

$$C = \frac{I_{tot}}{2}\left[(r - i - 1) + \sqrt{(r + i)^2 - 2(r - i) + 1}\right] \quad (22)$$

Inserting equation (22) into equation (15) we have:

$$P(r) = y_{max}\frac{I_{tot}\left[(r-i-1)+\sqrt{(r+i)^2-2(r-i)+1}\right]}{2K_e + I_{tot}\left[(r-i-1)+\sqrt{(r+i)^2-2(r-i)+1}\right]} \quad (23)$$

Taking the Taylor series approximation of equation (23) near $r = 0$, we have:

$$P(r) = y_{max}\frac{K_I}{K_e}\frac{I_{tot}}{I_{tot}+K_I}\left[r + \frac{I_{tot}}{I_{tot}+K_I}\left(\frac{I_{tot}}{I_{tot}+K_I} - \frac{K_I}{K_e}\right)\cdot r^2\right] \quad (24)$$

Assuming that $I_{tot} > K_I$, we can further simplify to:

$$P(r) = y_{max}\frac{K_I}{K_e}\left[r + \left(1 - \frac{K_I}{K_e}\right)\cdot r^2\right] \quad (25)$$

Thus, P is upward sloping (positive curvature) when $K_e > K_I$, and downward sloping when $K_e < K_I$.

IIE. Matlab code for flow cytometric FRET

To be used with accompanying .fig file (not provided).

```
function varargout = flowfretc(varargin)

% Edit the above text to modify the response to help flowfretc
% Last Modified by GUIDE v2.5 30-Jul-2015 14:07:39
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @flowfretc_OpeningFcn, ...
                   'gui_OutputFcn',  @flowfretc_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before flowfretc is made visible.
function flowfretc_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for flowfretc

figobjects = fieldnames(handles);
defaultcolor = [1 1 0.95];
for k=1:length(figobjects)
    if strncmp('axes',figobjects{k},4) & isempty(strfind(figobjects{k},'context'))
        set(handles.(figobjects{k}),'FontSize',6,'color',defaultcolor);
    end
    if ~strncmp('axes',figobjects{k},4) & isempty(strfind(figobjects{k},'context')) &
isempty(strfind(figobjects{k},'figure')) & ~strcmp(get(handles.(figobjects{k}),'type'),'uimenu') &
isempty(strfind(figobjects{k},'ui'))
        set(handles.(figobjects{k}),'BackgroundColor',defaultcolor);
    end
end
handles.output = hObject;

% set color scheme
FRcolor = [0.96 0.97 0.99];
EFRETcolor = [0.97 0.97 1.0];

FRobjects = {'numinroi_edit' 'axes4' 'axes5' 'axes6' 'axes7' 'KD' 'EFRmax' 'spurFR' 'xbin_edit' 'filter1_edit' 'bin'};
EFRETobjects = {'numinroi2_edit' 'axes8' 'axes9'  'KD2' 'EFRETmax' 'spurEFRET' 'xbin2_edit' 'filter2_edit' 'bin2'};

for k=1:length(FRobjects)
```

```
    if strncmp('axes',FRobjects{k},4)
        set(handles.(FRobjects{k}),'color',FRcolor);
    else
        set(handles.(FRobjects{k}),'BackgroundColor',FRcolor);
    end
end

for k=1:length(EFRETobjects)
    if strncmp('axes',EFRETobjects{k},4)
        set(handles.(EFRETobjects{k}),'color',EFRETcolor);
    else
        set(handles.(EFRETobjects{k}),'BackgroundColor',EFRETcolor);
    end
end


handles.dir = cd;
handles.reset = 0;

handles.axes4xlabel = 'S_C_F_P';
handles.axes4ylabel = 'S_Y_F_P';
handles.axes5xlabel = 'S_C_F_P';
handles.axes5ylabel = 'S_F_R_E_T';

% map = lines;
handles.roi_color = [[0,0,0] ; [0 0.5 0]; [0 0.2 0.2]; [1 0 1]; [0.5 0 0];...
                     [0 1 0] ; [0 0.5 0]; [1 0 0]; [1 0.5 0]; [0 0 1]];% generate 10 colors

set(gcf,'KeyPressFcn',@fighotkeys);

set(handles.keeprois,'Checked','off');
set(handles.showfiltered,'Checked','on');
set(handles.listbox,'FontSize',7);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes flowfretc wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = flowfretc_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in chooseanalysis.
function chooseanalysis_Callback(hObject, eventdata, handles)
% hObject    handle to chooseanalysis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.dir = uigetdir;
set(handles.listbox,'value',1);
set(handles.listbox,'string',{[]});
set(handles.construct_listbox,'value',1);
set(handles.construct_listbox,'string',{[]});
if isnumeric(handles.dir)
    return;
end
% cd(handles.dir);
%choose or create analysis file
choice = questdlg('Create new or choose existing analysis file?', ...
    '', ...
    'Create New','Choose Existing','Cancel','Cancel');
switch choice
    case 'Create New'
        FileName = inputdlg('Please enter filename (without extension)');
        FileName = FileName{1};
        FileName = [FileName, '.flow'];
        PathName = [handles.dir, '\'];
    case 'Choose Existing'
        [FileName,PathName,FilterIndex] = uigetfile([handles.dir '\*.flow']);
    case 'Cancel'
        return;
end
handles.analysisfile = [PathName,FileName];
set(handles.analysisfile_text,'string',FileName);

for k=1:9
    eval(['cla(handles.axes' num2str(k) ');'])
end

for k=1:3
    try
```

```
        eval(['handles = rmfield(handles, ''roi' num2str(k) 'poly'');']);
    end
end

switch choice
    case 'Create New'

        %create rudimentary file structure for analysis file
        DATA.datecreated = date;
        DATA.version = get(gcf,'name');



        save(handles.analysisfile,'-struct','DATA');
        fprintf('New analysis file created\n');

    case 'Choose Existing'
        %load DATA file and populate listbox
        DATA = load(handles.analysisfile,'-mat');
        try
            if ~strcmp(get(gcf,'name'),DATA.version)
                errordlg('Data structure was created with a different analysis software!');
            end
        catch  % older versions don't have DATA.(structname).analysisver
            errordlg('Data structure was created with a different analysis software!');
            return;
        end

        % populate construct_listbox
        for k=1:length(DATA.constructs)
            listboxstr{k} = DATA.constructs(k).name;
        end
        set(handles.construct_listbox,'string',listboxstr,'Max',length(listboxstr));

end

guidata(hObject,handles);

%%

% --- Executes on selection change in construct_listbox.
function construct_listbox_Callback(hObject, eventdata, handles)
if ~isfield(handles,'analysisfile')
    return
end

DATA = load(handles.analysisfile,'-mat');

clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');

if isempty(clistboxstr{1})
    return
end

linkaxes([handles.axes6 handles.axes7],'off');
linkaxes([handles.axes8 handles.axes9],'off');
listboxstr = [];
confilelength = [];
for k=1:length(clistboxval)
    conname = clistboxstr{clistboxval(k)};
    conindex = find(strcmp({DATA.constructs.name},conname));

    if isfield(DATA.constructs(conindex),'datafile')
        temp = {''};
        tempfilelength = 0;
        if ~isempty(DATA.constructs(conindex).datafile)
            tempfilelength = length(DATA.constructs(conindex).datafile);
            for k=1:length(DATA.constructs(conindex).datafile)
                temp{k} = DATA.constructs(conindex).datafile(k).name;
            end
        end
        set(handles.listbox,'string',listboxstr,'Value',1,'Max',length(listboxstr));
    end
    listboxstr = [listboxstr, temp];
    confilelength = [confilelength, tempfilelength];
end
handles.confilelength = confilelength;
set(handles.listbox,'string',listboxstr,'Value',1,'Max',length(listboxstr));
guidata(hObject,handles);

% --------------------------------------------------------------------
function addconstruct_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
conname = inputdlg('Please input construct name');
if isfield(DATA,'constructs')
    numconstructs = length(DATA.constructs);
else
    numconstructs = 0;
end
```

```
% initialize fields
DATA.constructs(numconstructs+1).name= conname{1};
DATA.constructs(numconstructs+1).datafile(1).name = [];
DATA.constructs(numconstructs+1).datafile(1).filename = [];
DATA.constructs(numconstructs+1).datafile(1).roi1 = [];
DATA.constructs(numconstructs+1).datafile(1).roi2 = [];
DATA.constructs(numconstructs+1).datafile(1).roi3 = [];
DATA.constructs(numconstructs+1).datafile(1).filter1 = [];
DATA.constructs(numconstructs+1).datafile(1).filter2 = [];
DATA.constructs(numconstructs+1).datafile(1).RA1 = [];
DATA.constructs(numconstructs+1).datafile(1).RA2 = [];
DATA.constructs(numconstructs+1).datafile(1).RA3 = [];
DATA.constructs(numconstructs+1).datafile(1).RD1l = [];
DATA.constructs(numconstructs+1).datafile(1).RD2l = [];
DATA.constructs(numconstructs+1).datafile(1).RD3 = [];
DATA.constructs(numconstructs+1).datafile(1).RC1 = [];
DATA.constructs(numconstructs+1).datafile(1).RC2 = [];
DATA.constructs(numconstructs+1).datafile(1).RC3 = [];
DATA.constructs(numconstructs+1).datafile(1).BG = [];
DATA.constructs(numconstructs+1).datafile(1).GAGD = [];
DATA.constructs(numconstructs+1).datafile(1).FAFD = [];
DATA.constructs(numconstructs+1).plottype = [];

for k=1:length(DATA.constructs)
    listboxstr{k} = DATA.constructs(k).name;
end
set(handles.construct_listbox,'string',listboxstr,'Value',length(listboxstr),'Max',length(listboxstr));

set(handles.listbox,'string',{''},'value',1);
save(handles.analysisfile,'-struct','DATA','-append');

% --------------------------------------------------------------------
function removeconstruct_Callback(hObject, eventdata, handles)
button = questdlg('Are you sure you want to remove this?','Confirm remove','Yes','No','No');
if strcmp(button,'No')
    return;
end
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
conname = listboxstr{listboxval};

todelete = find(strcmp({DATA.constructs.name},conname));
DATA.constructs(todelete) = [];

listboxstr = [];
if ~isempty(DATA.constructs)
    for k=1:length(DATA.constructs)
        listboxstr{k} = DATA.constructs(k).name;
    end
    listboxval = k;
else
    listboxstr = {''};
    listboxval = 1;
end
set(handles.construct_listbox,'string',listboxstr,'Value',listboxval,'Max',listboxval);
save(handles.analysisfile,'-struct','DATA','-append');
construct_listbox_Callback(handles.construct_listbox,[],handles);


% --------------------------------------------------------------------
function addfile_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));

if ~isempty(DATA.constructs(conindex).datafile(1).filename)
    datafilelength = length(DATA.constructs(conindex).datafile);
else
    datafilelength = 0;
end

[FileName,PathName,FilterIndex] = uigetfile([handles.dir '\*.fcs']);
if ~isempty(strfind(PathName,handles.dir))
    PathName = PathName(length(handles.dir)+1:end);
else
    errordlg('Please select file within the working directory');
    return;
end
DATA.constructs(conindex).datafile(datafilelength + 1).filename = [PathName,FileName];
inds = strfind(FileName,'default config ');
if ~isempty(inds)
    FileName(inds(1):inds(1)+14) = '';
end
dispname = inputdlg('Display as','',1,{FileName(1:end-4)});
DATA.constructs(conindex).datafile(datafilelength + 1).name = dispname{1};
listboxstr = [];
for k=1:length(DATA.constructs(conindex).datafile)
    listboxstr{k} = DATA.constructs(conindex).datafile(k).name;
```

```
end
set(handles.listbox,'string',listboxstr,'Max',length(listboxstr),'Value',length(listboxstr));
handles.confilelength = length(listboxstr);
save(handles.analysisfile,'-struct','DATA','-append');
guidata(hObject,handles);

% --------------------------------------------------------------------
function removefile_Callback(hObject, eventdata, handles)
button = questdlg('Are you sure you want to remove this?','Confirm remove','Yes','No','No');
if strcmp(button,'No')
    return;
end
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));

listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');
sel = listboxstr{listboxval};
todelete = find(strcmp({DATA.constructs(conindex).datafile.name},sel));
DATA.constructs(conindex).datafile(todelete) = [];

listboxstr = [];
if ~isempty(DATA.constructs(conindex).datafile)
    for k=1:length(DATA.constructs(conindex).datafile)
        listboxstr{k} = DATA.constructs(conindex).datafile(k).name;
    end
    listboxval = k;
else
    listboxstr = {''};
    listboxval = 1;
end
set(handles.listbox,'string',listboxstr,'Value',listboxval,'Max',length(listboxstr));
handles.confilelength = length(listboxstr);
save(handles.analysisfile,'-struct','DATA','-append');
guidata(hObject,handles);

% --------------------------------------------------------------------
function toggleaxis(hObject, eventdata, handles)
cp = get(gca,'CurrentPoint');
xlims = get(gca,'XLim');
if cp(1,1)<xlims(1)
    currentlabel = 'ylabel';
else
    currentlabel = 'xlabel';
end
currentaxes = get(gca,'tag');
switch get(hObject,'tag');
    case 'Scfpaxis'
        handles.([currentaxes, currentlabel]) = 'S_C_F_P';
    case 'Syfpaxis'
        handles.([currentaxes, currentlabel]) = 'S_Y_F_P';
    case 'Sfretaxis'
        handles.([currentaxes, currentlabel]) = 'S_F_R_E_T';
    case 'BL2axis'
        handles.([currentaxes, currentlabel]) = 'BL2';
    case 'BL3axis'
        handles.([currentaxes, currentlabel]) = 'BL3';

end
guidata(hObject,handles);
for k=4:5
    delete(findobj(handles.(['axes' num2str(k)]),'tag','data'));
end
UpdateFlowFretC;


% --------------------------------------------------------------------
function associatefiles_Callback(hObject, eventdata, handles)

DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));

switch DATA.constructs(conindex).plottype
    case 'BG'
        svar = 'BGi';
    case 'RA'
        svar = 'RAi';
    case 'RD'
        svar = 'RDi';
    case 'MAMD'
        svar = 'MAMDi';
```

```
    case 'RC'
        svar = 'RCi';
end

listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');
if strcmp(svar,'BGi')
    dates = (cellfun(@(x) x(1:10), listboxstr, 'UniformOutput', false));
    uniquedates = unique(cellstr(datestr(dates,'dd-mmm-yyyy')));
    [sorted,order] = sort(datenum(uniquedates));
    uniquedates = uniquedates(order);
else
    uniquedates = {DATA.constants.date};
end

% if ~strcmp(svar,'MAMDi')
%     for z=1:length(listboxstr)
%         sel = listboxstr{z};
%         fileindex = find(strcmp({DATA.constructs(conindex).datafile.name},sel));
%         filename = DATA.constructs(conindex).datafile(fileindex).filename;
%         [fcsdat, fcshdr] = fca_readfcs(filename);
%         date(z) = {fcshdr.date};
%     end
%     uniquedates = unique(date);
%     uniquedates2 = {DATA.constants.date};
%     if length(uniquedates2)>length(uniquedates)
%         uniquedates = uniquedates2;
%     end
% else
%     uniquedates = {DATA.constants.date};
% end
currfileindex = find(strcmp({DATA.constructs(conindex).datafile.name},listboxstr{listboxval}));
formatout = 'dd-mmm-yyyy';
sortdates = datestr(sort(datenum(uniquedates)),formatout);
[chosendate,v] = listdlg('PromptString','Associate with following date:',...
            'SelectionMode','single',...
            'ListString',sortdates);

if v
    chosendate = sortdates(chosendate,:);
    constantsindex = find(strcmp(chosendate,{DATA.constants.date}));

    if ~isfield(DATA.constructs(conindex),'plottype')
        errordlg('Set Plot Type first');
        return;
    else
        if isempty(DATA.constructs(conindex).plottype)
            errordlg('Set Plot Type first');
            return;
        end
    end
end


if v
    DATA.constants(constantsindex).(svar) = [conindex currfileindex];
end
boxstr(1) = {'Date              <--           file'};
for k=1:size(sortdates,1)
    chosendate = sortdates(k,:);
    constindex = find(strcmp(chosendate,{DATA.constants.date}));
    if ~isempty(DATA.constants(constindex).(svar))
        cindex = DATA.constants(constindex).(svar)(1);
        findex = DATA.constants(constindex).(svar)(2);
        fname = DATA.constructs(cindex).datafile(findex).name;
    else
        fname = 'No file associated';
    end
    boxstr(k+1) = {[chosendate ' <-- ' fname]};
end
msgbox(boxstr);
if v
    save(handles.analysisfile,'-struct','DATA','-append');
end
%%

function listbox_Callback(hObject, eventdata, handles)
for k=1:9
    eval(['delete(findobj(handles.axes' num2str(k) ', ''Tag'', ''data''));'])
    eval(['delete(findobj(handles.axes' num2str(k) ', ''type'',''text''));'])
end
filterheader = [ isfield(handles,'roi1poly'), isfield(handles,'roi2poly'), isfield(handles,'roi3poly')];
toplotfilt = find(filterheader);
handles.toplotfilt = toplotfilt;

%reset triggers
handles.tryfitFR1 = 0;
handles.tryfitFR2 = 0;
handles.tryfitEFRET1 = 0;
handles.tryfitEFRET2 = 0;
```

```
% parse conindex and fileindex
DATA = load(handles.analysisfile,'-mat');
clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');
confilelength = handles.confilelength;

listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');

for z=1:length(listboxval)
    handles.z = z;
    sel = listboxstr{listboxval(z)};
    cval = find(listboxval(z)<=cumsum(confilelength),1,'first');
    conname = clistboxstr{cval};
    conindex = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    fileindex = find(strcmp({DATA.constructs(conindex).datafile.name},sel));

    if strcmp(DATA.constructs(conindex).plottype,'MAMD')
        if isfield(DATA.constructs(conindex).datafile(fileindex),'filter1')
            set(handles.filter1_edit,'string',DATA.constructs(conindex).datafile(fileindex).filter1);
        else
            if strcmpi(get(handles.keeprois,'Checked'),'off')
                set(handles.filter1_edit,'string','');
            end
        end
        guidata(hObject,handles);
        setMAMD_Callback(handles.setMAMD,[],handles);
        guidata(hObject,handles);
        return;
    end
    filename = DATA.constructs(conindex).datafile(fileindex).filename;
    filenameindex = strfind(filename,'FLOW');   % hack to include older files from SRL's computer
    if ~isempty(filenameindex)
        filename = [handles.dir, filename(filenameindex+4:end)];
    else
        filename = [handles.dir, filename];
    end
    [fcsdat, fcshdr] = fca_readfcs(filename);

    if isempty(fcshdr)
        errordlg('Header file cannot be read');
        return;
    end
    %
    % define the parameter names
    %
    if strfind(fcshdr.cytometry,'Attune')
        handles.saturation = 1e7;
        handles.CFPname = 'VL1-A';
        handles.YFPname = 'BL1-A';
        handles.FRETname = 'VL2-A';
        handles.BL2name = 'BL2-A';
        handles.BL3name = 'BL3-A';
        handles.singlet1x = 'FSCA';
        handles.singlet1y = 'FSCH';
        handles.singlet2x = 'SSCA';
        handles.singlet2y = 'SSCH';
    else
        handles.saturation = 262000;
        handles.CFPname = 'pacific Blue-A';
        handles.YFPname = 'FITC-A';
        handles.FRETname = 'AmCyan-A';
        handles.singlet1x = 'FSCH';
        handles.singlet1y = 'FSCW';
        handles.singlet2x = 'SSCH';
        handles.singlet2y = 'SSCW';
    end

    set(handles.filename_editbox,'String',fcshdr.filename);
    parnames_long = cell(1,fcshdr.NumOfPar);
    parnames = cell(1,fcshdr.NumOfPar);
    for i=1:fcshdr.NumOfPar
        parnames(i) = {fcshdr.par(i).name};
    end
    handles.parnames = parnames;

    set(handles.numofevent_edit,'String',fcshdr.TotalEvents);
    set(handles.numinroi_edit,'String',fcshdr.TotalEvents);
    handles.TotalEvents = fcshdr.TotalEvents;
    if isempty(find(strcmpi(handles.CFPname,handles.parnames)))
        cfpvalue = listdlg('PromptString','Select CFP channel:',...
                    'SelectionMode','single',...
                    'ListString',handles.parnames);
        handles.CFPname = handles.parnames{cfpvalue};
        yfpvalue = listdlg('PromptString','Select YFP channel:',...
                    'SelectionMode','single',...
                    'ListString',handles.parnames);
        handles.YFPname = handles.parnames{yfpvalue};
        fretvalue = listdlg('PromptString','Select FRET channel:',...
                    'SelectionMode','single',...
                    'ListString',handles.parnames);
```

```
    handles.FRETname = handles.parnames{fretvalue};
else
    cfpvalue = find(strcmpi(handles.CFPname,handles.parnames));
    yfpvalue = find(strcmpi(handles.YFPname,handles.parnames));
    fretvalue = find(strcmpi(handles.FRETname,handles.parnames));
    BL2value = find(strcmpi(handles.BL2name,handles.parnames));
    BL3value = find(strcmpi(handles.BL3name,handles.parnames));
end
set(handles.cfpchan_popup,'string',handles.CFPname);
set(handles.yfpchan_popup,'string',handles.YFPname);
set(handles.fretchan_popup,'string',handles.FRETname);

set(handles.cfpPMT,'string',fcshdr.par(cfpvalue).PMTV);
set(handles.yfpPMT,'string',fcshdr.par(yfpvalue).PMTV);
set(handles.fretPMT,'string',fcshdr.par(fretvalue).PMTV);

handles.FSCPMTV = fcshdr.par(find(strcmpi('FSC-A',handles.parnames))).PMTV;
handles.SSCPMTV = fcshdr.par(find(strcmpi('SSC-A',handles.parnames))).PMTV;
handles.cfpPMTV = fcshdr.par(cfpvalue).PMTV;
handles.yfpPMTV = fcshdr.par(yfpvalue).PMTV;
handles.fretPMTV = fcshdr.par(fretvalue).PMTV;
handles.BL2PMTV = fcshdr.par(BL2value).PMTV;
handles.BL3PMTV = fcshdr.par(BL3value).PMTV;

handles.FSCA = fcsdat(:,find(strcmpi('FSC-A',handles.parnames)));
handles.SSCA = fcsdat(:,find(strcmpi('SSC-A',handles.parnames)));
handles.FSCH = fcsdat(:,find(strcmpi('FSC-H',handles.parnames)));
handles.FSCW = fcsdat(:,find(strcmpi('FSC-W',handles.parnames)));
handles.SSCH = fcsdat(:,find(strcmpi('SSC-H',handles.parnames)));
handles.SSCW = fcsdat(:,find(strcmpi('SSC-W',handles.parnames)));
handles.flowtime = fcsdat(:,1);
handles.cfp = fcsdat(:,cfpvalue);
handles.yfp = fcsdat(:,yfpvalue);
handles.fret = fcsdat(:,fretvalue);
handles.BL2 = fcsdat(:,BL2value);
handles.BL3 = fcsdat(:,BL3value);

handles.fcsdate = fcshdr.date;
% scale fluorescence values to the same PMTV for Attune
if strfind(fcshdr.cytometry,'Attune')
    load([handles.dir '\PMTcalib.mat']);
    % scale FSC
    defaultPMTV = 1750;
    currentPMTV = str2num(handles.FSCPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(FSCHc),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.FSCA = handles.FSCA*scalefactor;
    handles.FSCH = handles.FSCH*scalefactor;

    % scale SSC
    defaultPMTV = 2250;
    currentPMTV = str2num(handles.SSCPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(SSCHc),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.SSCA = handles.SSCA*scalefactor;
    handles.SSCH = handles.SSCH*scalefactor;

    % scale BL-1
    defaultPMTV = 1000;
    currentPMTV = str2num(handles.yfpPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(BL1Ac),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.yfp = handles.yfp*scalefactor;

    % scale VL-1
    defaultPMTV = 1000;
    currentPMTV = str2num(handles.cfpPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(VL1Ac),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.cfp = handles.cfp*scalefactor;

    % scale VL-2
    defaultPMTV = 1000;
    currentPMTV = str2num(handles.fretPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(VL2Ac),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.fret = handles.fret*scalefactor;

    % scale BL-2
    defaultPMTV = 1000;
    currentPMTV = str2num(handles.BL2PMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(BL2Ac),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.BL2 = handles.BL2*scalefactor;
```

```
        % scale BL-3
        defaultPMTV = 1500;
        currentPMTV = str2num(handles.BL3PMTV);
        xi = [currentPMTV defaultPMTV];
        yi = exp(interp1(log(PMTVc),log(BL3Ac),log(xi)));
        scalefactor = yi(2)/yi(1);
        handles.BL3 = handles.BL3*scalefactor;

    else
        warndlg('PMTVs are not scaled. Do not have PMT calibration data for this cytometer');
    end


    if isfield(DATA.constructs(conindex).datafile(fileindex),'roi1') &
~isempty(DATA.constructs(conindex).datafile(fileindex).roi1)
        handles.roi1pos = DATA.constructs(conindex).datafile(fileindex).roi1;
        if ~isfield(handles,'roi1poly')
            handles.roi1poly = impoly(handles.axes1,handles.roi1pos);
            setColor(handles.roi1poly,'red');
            addNewPositionCallback(handles.roi1poly,@roi1Update);
        else
                setPosition(handles.roi1poly,handles.roi1pos);
        end
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            if isfield(handles,'roi1poly')
                delete(handles.roi1poly);
                handles = rmfield(handles, 'roi1poly');
            end
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'roi2') &
~isempty(DATA.constructs(conindex).datafile(fileindex).roi2)
        handles.roi2pos = DATA.constructs(conindex).datafile(fileindex).roi2;
        if ~isfield(handles,'roi2poly')
            handles.roi2poly = impoly(handles.axes2,handles.roi2pos);
            setColor(handles.roi2poly,[0 1 0]);
            addNewPositionCallback(handles.roi2poly,@roi2Update);
        else
                setPosition(handles.roi2poly,handles.roi2pos);
        end
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            if isfield(handles,'roi2poly')
                delete(handles.roi2poly);
                handles = rmfield(handles, 'roi2poly');
            end
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'roi3') &
~isempty(DATA.constructs(conindex).datafile(fileindex).roi3)
        handles.roi3pos = DATA.constructs(conindex).datafile(fileindex).roi3;
        if ~isfield(handles,'roi3poly')
            handles.roi3poly = impoly(handles.axes3,handles.roi3pos);
            setColor(handles.roi3poly,[ 1 0 1 ]);
            addNewPositionCallback(handles.roi3poly,@roi3Update);
        else
                setPosition(handles.roi3poly,handles.roi3pos);
        end
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            if isfield(handles,'roi3poly')
                delete(handles.roi3poly);
                handles = rmfield(handles, 'roi3poly');
            end
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'roi4') &
~isempty(DATA.constructs(conindex).datafile(fileindex).roi4)
        handles.roi4pos = DATA.constructs(conindex).datafile(fileindex).roi4;
        if ~isfield(handles,'roi4poly')
            handles.roi4poly = impoly(handles.axes4,handles.roi4pos);
            setColor(handles.roi4poly,[ 1 0 1 ]);
            addNewPositionCallback(handles.roi4poly,@roi4Update);
        else
                setPosition(handles.roi4poly,handles.roi4pos);
        end
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            if isfield(handles,'roi4poly')
                delete(handles.roi4poly);
                handles = rmfield(handles, 'roi4poly');
            end
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'roi5') &
~isempty(DATA.constructs(conindex).datafile(fileindex).roi5)
        handles.roi5pos = DATA.constructs(conindex).datafile(fileindex).roi5;
        if ~isfield(handles,'roi5poly')
            handles.roi5poly = impoly(handles.axes5,handles.roi5pos);
            setColor(handles.roi5poly,[ 1 0 1 ]);
```

```
            addNewPositionCallback(handles.roi5poly,@roi5Update);
        else
                setPosition(handles.roi5poly,handles.roi5pos);
        end
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            if isfield(handles,'roi5poly')
                delete(handles.roi5poly);
                handles = rmfield(handles, 'roi5poly');
            end
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'filter1')
        set(handles.filter1_edit,'string',DATA.constructs(conindex).datafile(fileindex).filter1);
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            set(handles.filter1_edit,'string','');
        end
    end
    if isfield(DATA.constructs(conindex).datafile(fileindex),'filter2')
        set(handles.filter2_edit,'string',DATA.constructs(conindex).datafile(fileindex).filter2);
    else
        if strcmpi(get(handles.keeprois,'Checked'),'off')
            set(handles.filter2_edit,'string','');
        end
    end

    handles.conindex = conindex;
    handles.fileindex = fileindex;

    guidata(hObject,handles);
    drawnow;
    UpdateFlowFretC;
    drawnow;
    guidata(hObject,handles);
end



%%
function UpdateFlowFretC(startpos)
if nargin<1
    startpos = 1;
end

% startpos = 1 ( do everything )
% startpos = 2 ( skip singlet processing )
% startpos = 3 ( skip singlet and fluorescence filtering ) <--not employed
figh = findobj('type','figure');
if length(figh)>1
   for k=1:length(figh)
       fign = get(figh,'name');
       if ~isempty(fign{k})
           rfigh = figh(k);
       end
   end
   figh = rfigh;
end
handles = guidata(figh);

DATA = load(handles.analysisfile,'-mat');
clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');
confilelength = handles.confilelength;

listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');

z = handles.z;
sel = listboxstr{listboxval(z)};
cval = find(listboxval(z)<=cumsum(confilelength),1,'first');
connname = clistboxstr{cval};
conindex = clistboxval(find(strcmp({DATA.constructs.name},conname)));
% fileindex = handles.fileindex;
fileindex = find(strcmp({DATA.constructs(conindex).datafile.name},sel));
try
    TotalEvents = handles.TotalEvents;
end

if strcmp(DATA.constructs(conindex).plottype,'MAMD')
    filter1 = get(handles.filter1_edit,'string');
    DATA.constructs(conindex).datafile(fileindex).filter1 = filter1;
    save(handles.analysisfile,'-struct','DATA','-append');

    setMAMD_Callback(handles.setMAMD,[],handles);
    guidata(gcf,handles);
    return;
end

% first determine which filters are present
filter1 = get(handles.filter1_edit,'string');
```

```
filter2p = get(handles.filter2_edit,'string');
filterheader = [ isfield(handles,'roi1poly'), isfield(handles,'roi2poly'), isfield(handles,'roi3poly'),
isfield(handles,'roi4poly'), isfield(handles,'roi5poly'), ~isempty(filter1), ~isempty(filter2p)];
if startpos<3

    if filterheader(1)
        handles.roi1index = inpolygon(handles.FSCH,handles.SSCH,handles.roi1pos(:,1),handles.roi1pos(:,2));
        DATA.constructs(conindex).datafile(fileindex).roi1 = handles.roi1pos;
    else
        handles.roi1index = ones(TotalEvents,1);
        if isfield(DATA.constructs(conindex).datafile(fileindex),'roi1')
            DATA.constructs(conindex).datafile(fileindex).roi1 = [];
        end
    end
    if filterheader(2)
        handles.roi2index =
inpolygon(handles.(handles.singlet1x),handles.(handles.singlet1y),handles.roi2pos(:,1),handles.roi2pos(:,2));
        DATA.constructs(conindex).datafile(fileindex).roi2 = handles.roi2pos;
    else
        handles.roi2index = ones(TotalEvents,1);
        if isfield(DATA.constructs(conindex).datafile(fileindex),'roi2')
            DATA.constructs(conindex).datafile(fileindex).roi2 = [];
        end
    end
    if filterheader(3)
        handles.roi3index =
inpolygon(handles.(handles.singlet2x),handles.(handles.singlet2y),handles.roi3pos(:,1),handles.roi3pos(:,2));
        DATA.constructs(conindex).datafile(fileindex).roi3 = handles.roi3pos;
    else
        handles.roi3index = ones(TotalEvents,1);
        if isfield(DATA.constructs(conindex).datafile(fileindex),'roi3')
            DATA.constructs(conindex).datafile(fileindex).roi3 = [];
        end
    end
    xlab = handles.axes4xlabel;
    ylab = handles.axes4ylabel;
    xlab(find(xlab=='_')) = '';
    ylab(find(ylab=='_')) = '';
    xlab = lower(xlab(2:end));
    ylab = lower(ylab(2:end));
    if filterheader(4)
        handles.roi4index = ~inpolygon(handles.(xlab),handles.(ylab),handles.roi4pos(:,1),handles.roi4pos(:,2));
        DATA.constructs(conindex).datafile(fileindex).roi4 = handles.roi4pos;
    else
        handles.roi4index = ones(TotalEvents,1);
        if isfield(DATA.constructs(conindex).datafile(fileindex),'roi4')
            DATA.constructs(conindex).datafile(fileindex).roi4 = [];
        end
    end

    xlab = handles.axes5xlabel;
    ylab = handles.axes5ylabel;
    xlab(find(xlab=='_')) = '';
    ylab(find(ylab=='_')) = '';
    xlab = lower(xlab(2:end));
    ylab = lower(ylab(2:end));

    if filterheader(5)
        handles.roi5index = ~inpolygon(handles.(xlab),handles.(ylab),handles.roi5pos(:,1),handles.roi5pos(:,2));
        DATA.constructs(conindex).datafile(fileindex).roi5 = handles.roi5pos;
    else
        handles.roi5index = ones(TotalEvents,1);
        if isfield(DATA.constructs(conindex).datafile(fileindex),'roi5')
            DATA.constructs(conindex).datafile(fileindex).roi5 = [];
        end
    end

    % written filters
    cfp = handles.cfp;
    yfp = handles.yfp;
    fret = handles.fret;
    BL2 = handles.BL2;
    BL3 = handles.BL3;

    if filterheader(6)
        handles.filter1index = eval(filter1);
%         DATA.(structname).roi4 = handles.filter1index;
        DATA.constructs(conindex).datafile(fileindex).filter1 = filter1;
    else
        handles.filter1index = ones(TotalEvents,1);
%         if isfield(DATA.(structname),'roi4')
%             DATA.(structname) = rmfield(DATA.(structname),'roi4');
%         end

        DATA.constructs(conindex).datafile(fileindex).filter1 = [];

    end

    if filterheader(7)
        handles.filter2index = eval(filter2p);
%         DATA.(structname).roi5 = handles.filter2index;
```

```
        DATA.constructs(conindex).datafile(fileindex).filter2 = filter2p;
    else
        handles.filter2index = ones(TotalEvents,1);
%         if isfield(DATA.(structname),'roi5')
%             DATA.(structname) = rmfield(DATA.(structname),'roi5');
%         end
        DATA.constructs(conindex).datafile(fileindex).filter2 = [];
    end

    toplotfilt = find(filterheader);
    toplotfilt = [1 toplotfilt+1];
    handles.toplotfilt = toplotfilt;

    % do not display saturated and non-zero events
    nonsatnonzero = (handles.FSCA<handles.saturation & handles.SSCA<handles.saturation & handles.FSCA>0 &
handles.SSCA>0);
    nonsatnonzero = nonsatnonzero & cfp<handles.saturation & yfp<handles.saturation & fret<handles.saturation &
BL2<handles.saturation & BL3<handles.saturation & ...
            cfp>0 & yfp>0 & fret>0 & BL2>0 & BL3>0;
    singlets = handles.roi1index & handles.roi2index & handles.roi3index;
    filtered1 = nonsatnonzero & handles.roi1index & handles.roi2index & handles.roi3index ...
        & handles.roi4index & handles.roi5index & handles.filter1index;
    filtered2 = nonsatnonzero & handles.roi1index & handles.roi2index & handles.roi3index ...
        & handles.roi4index & handles.roi5index & handles.filter2index;

    set(handles.numinroi_edit,'string',num2str(sum(filtered1)));
    set(handles.numinroi2_edit,'string',num2str(sum(filtered2)));

    map = jet;
    z = handles.z;
    sel = get(handles.listbox,'Value');
    if length(sel)>1
        color = map(round(1+(z-1)/(length(sel)-1)*(length(map)-1)),:);
    else
        color = 'k';
    end

    if startpos<2
        axes(handles.axes1);
        xlabel('FSC-H'); ylabel('SSC-H');

        x = handles.FSCH(find(nonsatnonzero));
        y = handles.SSCH(find(nonsatnonzero));

        % only display max 10,000 points
        maxpts = 1e4;
        if length(x)<maxpts
            newx = x;
            newy = y;
%             line(x,y,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
        else
            ind = 1:ceil(length(x)/maxpts):length(x);
            newx = x(ind);
            newy = y(ind);
        end
        xlims = [0 5e6];
        ylims = [0 6e5];
%         dx = diff(xlims)*0.01;
%         dy = diff(ylims)*0.01;
%         for Z=1:length(newx)
%             roi = [newx(Z)-dx  newy(Z)-dy; ...
%                    newx(Z)-dx  newy(Z)+dy; ...
%                    newx(Z)+dx  newy(Z)+dy; ...
%                    newx(Z)+dx  newy(Z)-dy];
%             datainroi = inpolygon(newx,newy,roi(:,1),roi(:,2));
%             numpts(Z) = length(find(datainroi));
%         end
%         scatter(newx,newy,ones(size(newx)),numpts);
        line(newx,newy,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');

        if length(sel)==1
            [bandwidth,density,X,Y]=kde2d([x,y],2^8,[xlims(1),ylims(1)],[xlims(2) ylims(2)]);
            hold on;
            h = fspecial('gaussian',[20 20], 10);
            densitys = filter2(h,density); % smooth contours
            contour(X,Y,densitys,20,'tag','data');


%             contour(X,Y,densitys,20,'tag','data');
%             colormap(jet);
%             contour(X,Y,densitys,linspace(maxintensity*0.01,maxintensity*20,20),'tag','data');

            % set imroi on top
            lineh = get(gca,'Children');
            imh = find(strcmp(get(lineh,'tag'),'impoly'));
            if ~isempty(imh)
                if imh~=length(lineh)
                    newimh = [lineh(imh) ; lineh(1:imh-1); lineh(imh+1:end)];
                else
                    newimh = [lineh(imh) ; lineh(1:imh-1)];
                end
```

```
            set(gca,'Children',newimh);
        end
    end

    set(handles.axes1,'xlim',xlims,'ylim',ylims);

    axes(handles.axes2);
    xlabel(handles.singlet1x); ylabel(handles.singlet1y);
    x = handles.(handles.singlet1x)(find(nonsatnonzero & handles.roi1index));
    y = handles.(handles.singlet1y)(find(nonsatnonzero & handles.roi1index));
    if length(x)<maxpts
%         line(x,y,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
        newx = x;
        newy = y;
    else
        ind = 1:ceil(length(x)/maxpts):length(x);
        newx = x(ind);
        newy = y(ind);
    end
    xlims = [0 8e6];
    ylims = [0.3e6 4e6];

    line(newx,newy,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');

    if length(sel)==1
        [bandwidth,density,X,Y]=kde2d([x,y],2^8,[xlims(1) ylims(1)],[xlims(2),ylims(2)]);
        hold on;
        h = fspecial('gaussian',[20 20], 10);
        densitys = filter2(h,density);
        contour(X,Y,densitys,10,'tag','data');

        % set imroi on top
        lineh = get(gca,'Children');
        imh = find(strcmp(get(lineh,'tag'),'impoly'));
        if ~isempty(imh)
            if imh~=length(lineh)
                newimh = [lineh(imh) ; lineh(1:imh-1); lineh(imh+1:end)];
            else
                newimh = [lineh(imh) ; lineh(1:imh-1)];
            end
            set(gca,'Children',newimh);
        end
    end
    set(gca,'xlim',xlims,'ylim',ylims);

    axes(handles.axes3);
    xlabel(handles.singlet2x); ylabel(handles.singlet2y);

    x = handles.(handles.singlet2x)(find(nonsatnonzero & handles.roi1index));
    y = handles.(handles.singlet2y)(find(nonsatnonzero & handles.roi1index));

    if length(x)<maxpts
        newx = x;
        newy = y;
    else
        ind = 1:ceil(length(x)/maxpts):length(x);
        newx = x(ind);
        newy = y(ind);
    end
    xlims = [0 8e5];
    ylims = [0.5e5 5e5];

    line(newx,newy,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
    if length(sel)==1

        [bandwidth,density,X,Y]=kde2d([x,y],2^8,[xlims(1) ylims(1)],[xlims(2),ylims(2)]);
        hold on;
        h = fspecial('gaussian',[20 20], 10);
        densitys = filter2(h,density);
        contour(X,Y,densitys,10,'tag','data');
        % set imroi on top
        lineh = get(gca,'Children');
        imh = find(strcmp(get(lineh,'tag'),'impoly'));
        if ~isempty(imh)
            if imh~=length(lineh)
                newimh = [lineh(imh) ; lineh(1:imh-1); lineh(imh+1:end)];
            else
                newimh = [lineh(imh) ; lineh(1:imh-1)];
            end
            set(gca,'Children',newimh);
        end

        xlim(xlims);
        ylim(ylims);
    end
end

guidata(figh,handles);

end
%%
```

```
cfp = handles.cfp(find(filtered1));
yfp = handles.yfp(find(filtered1));
fret = handles.fret(find(filtered1));
BL2 = handles.BL2(find(filtered1));
BL3 = handles.BL3(find(filtered1));


if ~strcmpi(get(handles.showExp,'Checked'),'on')
%
    axes(handles.axes4);
    set(handles.axes4,'XScale','log','YScale','log');

    hx = xlabel(handles.axes4xlabel); hy = ylabel(handles.axes4ylabel);
    switch handles.axes4xlabel
        case 'S_C_F_P'
            x = cfp;
        case 'S_Y_F_P'
            x = yfp;
        case 'S_F_R_E_T'
            x = fret;
        case 'BL2'
            x = BL2;
        case 'BL3'
            x = BL3;
    end
    switch handles.axes4ylabel
        case 'S_C_F_P'
            y = cfp;
        case 'S_Y_F_P'
            y = yfp;
        case 'S_F_R_E_T'
            y = fret;
        case 'BL2'
            y = BL2;
        case 'BL3'
            y = BL3;
    end

    set(hx,'UIContextMenu',handles.axisselect_context);
    set(hy,'UIContextMenu',handles.axisselect_context);
    % only display max 20,000 points
    maxpts = 2e4;
    if length(x)<maxpts
        line(x,y,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
    else
        ind = 1:ceil(length(x)/maxpts):length(x);
        line(x(ind),y(ind),'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
    end
    if length(sel)==1
        xlim('auto'); ylim('auto');
    else
        xlim([1 1e7]); ylim([1 1e7]);
    end

    axes(handles.axes5);
    set(handles.axes5,'XScale','log','YScale','log');

    hx = xlabel(handles.axes5xlabel); hy = ylabel(handles.axes5ylabel);
    switch handles.axes5xlabel
        case 'S_C_F_P'
            x = cfp;
        case 'S_Y_F_P'
            x = yfp;
        case 'S_F_R_E_T'
            x = fret;
        case 'BL2'
            x = BL2;
        case 'BL3'
            x = BL3;
    end
    switch handles.axes5ylabel
        case 'S_C_F_P'
            y = cfp;
        case 'S_Y_F_P'
            y = yfp;
        case 'S_F_R_E_T'
            y = fret;
        case 'BL2'
            y = BL2;
        case 'BL3'
            y = BL3;
    end

    set(hx,'UIContextMenu',handles.axisselect_context);
    set(hy,'UIContextMenu',handles.axisselect_context);
    % only display max 20,000 points
    maxpts = 2e4;
    if length(x)<maxpts
        line(x,y,'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
```

```
    else
        ind = 1:ceil(length(x)/maxpts):length(x);
        line(x(ind),y(ind),'linestyle','none','Marker','.','color',color,'MarkerSize',1,'Tag','data');
    end
    if length(sel)==1
        xlim('auto'); ylim('auto');
    else
        xlim([1 1e7]); ylim([1 1e7]);
    end
end

% create structure for constants, one for each date.
if ~isfield(DATA,'constants')
    DATA.constants.date = {};
    DATA.constants.BGi = [];
    DATA.constants.RDi = [];
    DATA.constants.RAi = [];
    DATA.constants.RCi = [];
    DATA.constants.MAMDi = [];
    fieldlength = 0;
else
    fieldlength = length(DATA.constants);
end
date = handles.fcsdate;
constantsindex = find(strcmp(date,{DATA.constants.date}));
if isempty(constantsindex)
    DATA.constants(fieldlength+1).date = date;
    constantsindex = fieldlength+1;
end

linkaxes([handles.axes6 handles.axes7],'off');
linkaxes([handles.axes8 handles.axes9],'off');


if isfield(DATA.constructs(conindex),'plottype')
    if ~isempty(DATA.constructs(conindex).plottype)
        switch DATA.constructs(conindex).plottype
            case 'BG'
                BG(1) = trimmean(cfp,1);
                BG(2) = trimmean(yfp,1);
                BG(3) = trimmean(fret,1);
                BG(4) = trimmean(BL2,1);
                BG(5) = trimmean(BL3,1);
                BG(6) = std(cfp);
                BG(7) = std(yfp);
                BG(8) = std(fret);
                BG(9) = std(BL2);
                BG(10) = std(BL3);

                axes(handles.axes4);
                switch handles.axes4xlabel
                    case 'S_C_F_P'
                        line([BG(1) BG(1)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'S_Y_F_P'
                        line([BG(2) BG(2)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'S_F_R_E_T'
                        line([BG(3) BG(3)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'BL2'
                        line([BG(4) BG(4)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'BL3'
                        line([BG(5) BG(5)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');

                end
                switch handles.axes4ylabel
                    case 'S_C_F_P'
                        line(get(gca,'xlim'),[BG(1) BG(1)],'linestyle',':','color','k','Tag','data');
                    case 'S_Y_F_P'
                        line(get(gca,'xlim'),[BG(2) BG(2)], 'linestyle',':','color','k','Tag','data');
                    case 'S_F_R_E_T'
                        line(get(gca,'xlim'),[BG(3) BG(3)], 'linestyle',':','color','k','Tag','data');
                    case 'BL2'
                        line(get(gca,'xlim'),[BG(4) BG(4)], 'linestyle',':','color','k','Tag','data');
                    case 'BL3'
                        line(get(gca,'xlim'),[BG(5) BG(5)], 'linestyle',':','color','k','Tag','data');

                end
                axes(handles.axes5);
                switch handles.axes5xlabel
                    case 'S_C_F_P'
                        line([BG(1) BG(1)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'S_Y_F_P'
                        line([BG(2) BG(2)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'S_F_R_E_T'
                        line([BG(3) BG(3)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'BL2'
                        line([BG(4) BG(4)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');
                    case 'BL3'
                        line([BG(5) BG(5)], get(gca,'ylim'),'linestyle',':','color','k','Tag','data');

                end
                switch handles.axes5ylabel
```

```
                    case 'S_C_F_P'
                        line(get(gca,'xlim'),[BG(1) BG(1)],'linestyle',':','color','k','Tag','data');
                    case 'S_Y_F_P'
                        line(get(gca,'xlim'),[BG(2) BG(2)], 'linestyle',':','color','k','Tag','data');
                    case 'S_F_R_E_T'
                        line(get(gca,'xlim'),[BG(3) BG(3)], 'linestyle',':','color','k','Tag','data');
                    case 'BL2'
                        line(get(gca,'xlim'),[BG(4) BG(4)], 'linestyle',':','color','k','Tag','data');
                    case 'BL3'
                        line(get(gca,'xlim'),[BG(5) BG(5)], 'linestyle',':','color','k','Tag','data');

                end
                DATA.constructs(conindex).datafile(fileindex).BG = BG;
%%
            case 'RA'
                axes(handles.axes6);
                set(handles.axes6,'XScale','lin');

%                 ssca = handles.SSCA(find(filtered1));
                cfp = handles.cfp(find(filtered1));
                yfp = handles.yfp(find(filtered1));
                fret = handles.fret(find(filtered1));
                BL2 = handles.BL2(find(filtered1));
                BL3 = handles.BL3(find(filtered1));

                if ~isfield(DATA.constants(constantsindex),'BGi')
                    DATA.constants(constantsindex).BGi = [];
                end
                if isempty(DATA.constants(constantsindex).BGi)
                    cindex = find(strcmp('BG',{DATA.constructs.plottype}));
                    if ~isempty(cindex)
                        [findex,v] = listdlg('PromptString','Select a file for BG:',...
                            'SelectionMode','single',...
                            'ListString',{DATA.constructs(cindex).datafile.name});
                        figure(figh);
                        DATA.constants(constantsindex).BGi = [cindex findex];
                    else
                        errordlg('Please set BG first');
                        return;
                    end
                else
                    cindex = DATA.constants(constantsindex).BGi(1);
                    findex = DATA.constants(constantsindex).BGi(2);
                end
                BG = DATA.constructs(cindex).datafile(findex).BG;
                cfpb = cfp - BG(1);
                yfpb = yfp - BG(2);
                fretb = fret - BG(3);
                BL2b = BL2 - BG(4);
                BL3b = BL3 - BG(5);

                % bin, then fit binned data
                xbin = [-1e4:0.5e4:5e5, 5.1e5:2e4:1e6, 1.2e6:0.2e6:5e6];
                n = 1;
                for m=1:length(xbin)-1
                    indices = find(yfpb>=xbin(m) & yfpb<=xbin(m+1));
                    if length(indices)>2
%                         bmeanx(n) = exp(mean(log(yfpb(indices))));
%                         bmeany(n) = exp(mean(log(fretb(indices))));
                        bmeanx(n) = median(yfpb(indices));
                        bmeany(n) = median(fretb(indices));
                        bmeany2(n) = median(BL3b(indices));
                        n = n+1;
                    end
                end
                line(yfpb, fretb,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
%                 line(bmeanx,bmeany,'linestyle','none','Marker','.','MarkerSize',20,'tag','data');
                % two fits, one for Syfp<=5e5, another for Syfp>5e5
                thres = 3e5;
                indices1 = find(bmeanx<=thres);
%                 p1 = polyfit(bmeanx(indices1),bmeany(indices1),3);
%                 p2 = polyfit(bmeanx,bmeany,4);

                % also linear fit through origin
%                 indices1 = find(bmeanx<=thres);
                p4 = polyfit(bmeanx(indices1),bmeany(indices1),1);
                p5 = polyfit(bmeany2(indices1),bmeany(indices1),1);
%                 p4 = [bmeanx(indices1)'\bmeany(indices1)', 0];
%                 p5 = [bmeanx(indices1)'\bmeany2(indices1)', 0];

                xxx = linspace(0,thres,100);
                f = polyval(p4,xxx);
                line(xxx,f,'color','r','linewidth',2,'tag','data');
%                 xxx = linspace(thres,5e6,100);
%                 f = polyval(p2,xxx);
%                 line(xxx,f,'color','r','linewidth',2,'tag','data');
                xlabel('S_Y_F_P','FontName','Arial');ylabel('S_F_R_E_T','FontName','Arial');
                set(gca,'xlim',[0 thres],'ylim',[0 max(f)*5],'FontName','Arial');

                axes(handles.axes8);
                set(gca,'XScale','lin');
```

```
                line(BL3b, fretb,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
%                 line(bmeany2,bmeany,'linestyle','none','Marker','.','MarkerSize',20,'tag','data');

                xxx = linspace(0,20000,100);
                f = polyval(p5,xxx);
                line(xxx,f,'color','r','linewidth',2,'tag','data');
                xlabel('BL3','FontName','Arial');ylabel('S_F_R_E_T','FontName','Arial');
%                 set(gca,'xlim',[0 10000],'ylim',[0 max(f)*1.1],'FontName','Arial');
                set(gca,'xlim',[0 20000],'ylim',[0 1000],'FontName','Arial');

                if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RAl')
                    RAl = [];
                    RA2 = [];
                else
                    RAl = DATA.constructs(conindex).datafile(fileindex).RAl;
                    RA2 = DATA.constructs(conindex).datafile(fileindex).RA2;
                end
                if isempty(RAl) || isempty(RA2)
                    DATA.constructs(conindex).datafile(fileindex).RAl = p4;
%                     DATA.constructs(conindex).datafile(fileindex).RAh = p2;
                    DATA.constructs(conindex).datafile(fileindex).RAthres = thres;
                    DATA.constructs(conindex).datafile(fileindex).RA2 = p5;
                    set(handles.RA_edit,'string',num2str(p4(1)));
                    set(handles.RA2_edit,'string',num2str(p5(1)));
                else
                    set(handles.RA_edit,'string',num2str(RAl(1)));
                    set(handles.RA2_edit,'string',num2str(RA2(1)));
                    xxx = linspace(0,thres);
                    f = polyval(RAl,xxx);
                    axes(handles.axes6);
                    line(xxx,f,'color','m','linewidth',2,'tag','data');
                    f = polyval(RA2,xxx);
                    axes(handles.axes8);
                    line(xxx,f,'color','m','linewidth',2,'tag','data');

                end
%                 DATA.constructs(conindex).datafile(fileindex).RAl = p4;
%                 DATA.constructs(conindex).datafile(fileindex).RAh = p2;
%                 DATA.constructs(conindex).datafile(fileindex).RAthres = thres;
                axes(handles.axes6);
                text(0.1, 0.8, ['RA = ' num2str(p4(1))],'Units','Normalized');
                axes(handles.axes7);
                linkaxes([handles.axes6,handles.axes7],'x');

                underthres = find(yfpb<=thres);
                overthres = find(yfpb>thres);
%                 fity1 = polyval(p1,yfpb(underthres));
%                 fity2 = polyval(p2,yfpb(overthres));
                fity4 = polyval(p4,yfpb(underthres));
%                 errory = [(fity1-fretb(underthres)); ...
%                     fity2-fretb(overthres)];
                errory = (fity4-fretb(underthres))./fity4;

            line(yfpb(underthres),errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
                ylim([-1 1]);
                line(get(gca,'xlim'),[0,0],'Linestyle',':','LineWidth',5,'color','r','tag','data');

                axes(handles.axes8);
                text(0.1, 0.8, ['RA2 = ' num2str(p5(1))],'Units','Normalized');
                axes(handles.axes9);
                linkaxes([handles.axes8,handles.axes9],'x');

                fity4 = polyval(p5,BL3b);
                errory = (fity4-fretb)./fity4;
                line(BL3b,errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
                ylim([-1 1]);
                line(get(gca,'xlim'),[0,0],'Linestyle',':','LineWidth',5,'color','r','tag','data');

            case 'RD'
                axes(handles.axes6);
                set(handles.axes6,'XScale','lin');

%                 ssca = handles.SSCA(find(filtered1));
                cfp = handles.cfp(find(filtered1));
                yfp = handles.yfp(find(filtered1));
                fret = handles.fret(find(filtered1));
                BL3 = handles.BL3(find(filtered1));

                if ~isfield(DATA.constants(constantsindex),'BGi')
                    DATA.constants(constantsindex).BGi = [];
                end
                if isempty(DATA.constants(constantsindex).BGi)
                    cindex = find(strcmp('BG',{DATA.constructs.plottype}));
                    if ~isempty(cindex)
                        [findex,v] = listdlg('PromptString','Select a file for BG:',...
                            'SelectionMode','single',...
                            'ListString',{DATA.constructs(cindex).datafile.name});
                        figure(figh);
                        DATA.constants(constantsindex).BGi = [cindex findex];
                    else
                        errordlg('Please set BG first');
```

```
                    return;
                end
            else
                cindex = DATA.constants(constantsindex).BGi(1);
                findex = DATA.constants(constantsindex).BGi(2);
            end
            BG = DATA.constructs(cindex).datafile(findex).BG;
            cfpb = cfp - BG(1);
            yfpb = yfp - BG(2);
            fretb = fret - BG(3);
            BL2b = BL2 - BG(4);
            BL3b = BL3 - BG(5);

            % bin, then fit binned data
            xbin = [-1e4:0.5e4:1e5, 1.1e5:2e4:2e6]; %

            n = 1;
            for m=1:length(xbin)-1
                indices = find(cfpb>=xbin(m) & cfpb<=xbin(m+1));
                if length(indices)>2
%                     bmeanx(n) = exp(mean(log(yfpb(indices))));
%                     bmeany(n) = exp(mean(log(fretb(indices))));
                    bmeanx(n) = median(cfpb(indices));
                    bmeany(n) = median(fretb(indices));
                    bmeany2(n) = median(yfpb(indices));
                    bmeany3(n) = median(BL3b(indices));
                    n = n+1;
                end
            end
            line(cfpb, fretb,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
%             line(bmeanx,bmeany,'linestyle','none','Marker','.','MarkerSize',20,'tag','data');
            % two fits
            thres = 1e5;
            indices1 = find(bmeanx<=thres);
            p1 = polyfit(bmeanx(indices1),bmeany(indices1),3);
            p2 = polyfit(bmeanx,bmeany,4);

            % also try just fitting raw data
            ind1 = find(cfpb<=thres);
            p3 = polyfit(cfpb(ind1),fretb(ind1),1);
            xxx3 = linspace(0,thres);
            f3 = polyval(p3,xxx3);
%             line(xxx3,f3,'color','r','linewidth',2,'tag','data');

            % also try fit through origin
%             p4 = [cfpb(ind1)\fretb(ind1),0];
            p4 = polyfit(cfpb(ind1),fretb(ind1),1);
            f4 = polyval(p4,xxx3);
            line(xxx3,f4,'color','r','linewidth',2,'tag','data');
            xlabel('S_C_F_P','FontName','Arial');ylabel('S_F_R_E_T','FontName','Arial');
            set(gca,'xlim',[0 thres],'ylim',[0 max(f3)*1.1],'FontName','Arial');

            if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RD1l')
                RD1l = [];
            else
                RD1l = DATA.constructs(conindex).datafile(fileindex).RD1l;
            end
            if isempty(RD1l)
                DATA.constructs(conindex).datafile(fileindex).RD1l = p4;
                DATA.constructs(conindex).datafile(fileindex).RD1h = p2;
                DATA.constructs(conindex).datafile(fileindex).RD1thres = thres;
                set(handles.RD1_edit,'string',num2str(p4(1)));
                RD1l = p4;
            else
                set(handles.RD1_edit,'string',num2str(RD1l(1)));
                f = polyval(RD1l,xxx3);
                line(xxx3,f,'color','m','linewidth',2,'tag','data');
            end

            text(0.1, 0.8, ['RD1 = ' num2str(p4(1))],'Units','Normalized');
            axes(handles.axes5);
%             line([xxx1, xxx2]+BG(1),[f1 f2]+BG(3),'color','r','linewidth',2,'tag','data');
            line(xxx3+BG(1),f3+BG(3),'color','r','linewidth',2,'tag','data');

            axes(handles.axes7);
            linkaxes([handles.axes6,handles.axes7],'x');

            underthres = find(cfpb<=thres);
            overthres = find(cfpb>thres);
            fity1 = polyval(p1,cfpb(underthres));
            fity2 = polyval(p2,cfpb(overthres));
            fity3 = polyval(p3,cfpb(underthres));
            fity4 = polyval(p4,cfpb(underthres));
%             errory = [(fity1-fretb(underthres)); ...
%                 fity2-fretb(overthres)];
%             errory = (fity3-fretb(underthres))./fity3;
            errory = (fity4-fretb(underthres))./fity4;

line(cfpb(underthres),errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
            ylim([-1 1]);
            line(get(gca,'xlim'),[0,0],'Linestyle',':','color','r','tag','data');
```

```
            %%%%%%%%%

            axes(handles.axes8);
            line(cfpb, yfpb,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
%             line(bmeanx,bmeany2,'linestyle','none','Marker','.','MarkerSize',20,'tag','data');

            p4 = [cfpb(underthres)\yfpb(underthres) 0];
            f4 = polyval(p4,xxx3);
            line(xxx3,f4,'color','r','linewidth',2,'tag','data');
            text(0.1, 0.8, ['RD2 = ' num2str(p4(1))],'Units','Normalized');
            xlabel('S_C_F_P','FontName','Arial');ylabel('S_Y_F_P','FontName','Arial');
            set(gca,'xlim',[0 thres],'ylim',[0 max(f4)*1.1],'FontName','Arial','XScale','lin','YScale','lin');

            if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RD2l')
                RD2l = [];
            else
                RD2l = DATA.constructs(conindex).datafile(fileindex).RD2l;
            end
            if isempty(RD2l)
                DATA.constructs(conindex).datafile(fileindex).RD2l = p4;
                DATA.constructs(conindex).datafile(fileindex).RD2h = p2;
                set(handles.RD2_edit,'string',num2str(p4(1)));
                RD2l = p4;
            else
                set(handles.RD2_edit,'string',num2str(RD2l(1)));
                f = polyval(RD2l,xxx3);
                line(xxx3,f,'color','m','linewidth',2,'tag','data');
            end
%             DATA.constructs(conindex).datafile(fileindex).RD2thres = thres;

            axes(handles.axes9);
            linkaxes([handles.axes8,handles.axes9],'x');

            underthres = find(cfpb<=thres);
            overthres = find(cfpb>thres);
%             fity1 = polyval(p1,cfpb(underthres));
%             fity2 = polyval(p2,cfpb(overthres));
            fity4 = polyval(p4,cfpb(underthres));
%             errory = [(fity1-yfpb(underthres)); ...
%                 fity2-yfpb(overthres)];
            errory = (fity4-yfpb(underthres))./fity4;
            line(cfpb(ind1),errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
            ylim([-1 1]);
            line(get(gca,'xlim'),[0,0],'Linestyle',':','color','r','tag','data');

            %%%%%%

            axes(handles.axes8);
            line(cfpb, BL3b,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
            % also try fit through origin
%             p4 = polyfit(cfpb(ind1),BL3b(ind1),1);
            p4 = [cfpb(underthres)\BL3b(underthres) 0];
            f4 = polyval(p4,xxx3);
            line(xxx3,f4,'color','r','linewidth',2,'tag','data');
            text(0.1, 0.7, ['RD3 = ' num2str(p4(1))],'Units','Normalized');

            if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RD3')
                RD3 = [];
            else
                RD3 = DATA.constructs(conindex).datafile(fileindex).RD3;
            end
            if isempty(RD3)
                DATA.constructs(conindex).datafile(fileindex).RD3 = p4;
                set(handles.RD3_edit,'string',num2str(p4(1)));
            else
                set(handles.RD3_edit,'string',num2str(RD3(1)));
                f = polyval(RD3,xxx3);
                line(xxx3,f,'color','m','linewidth',2,'tag','data');
            end
%             DATA.constructs(conindex).datafile(fileindex).RD2thres = thres;

            axes(handles.axes9);
            linkaxes([handles.axes8,handles.axes9],'x');

            underthres = find(cfpb<=thres);
            fity4 = polyval(p4,cfpb(underthres));
            errory = (fity4-BL3b(underthres))./fity4;
            line(cfpb(ind1),errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
            ylim([-1 1]);
            line(get(gca,'xlim'),[0,0],'Linestyle',':','color','r','tag','data');
%%
        case 'RC'
            axes(handles.axes6);
            set(handles.axes6,'XScale','lin');

            cfp = handles.cfp(find(filtered1));
            yfp = handles.yfp(find(filtered1));
            fret = handles.fret(find(filtered1));
            BL2 = handles.BL2(find(filtered1));
            BL3 = handles.BL3(find(filtered1));
```

```
if ~isfield(DATA.constants(constantsindex),'BGi')
    DATA.constants(constantsindex).BGi = [];
end
if isempty(DATA.constants(constantsindex).BGi)
    cindex = find(strcmp('BG',{DATA.constructs.plottype}));
    if ~isempty(cindex)
        [findex,v] = listdlg('PromptString','Select a file for BG:',...
            'SelectionMode','single',...
            'ListString',{DATA.constructs(cindex).datafile.name});
        figure(figh);
        DATA.constants(constantsindex).BGi = [cindex findex];
    else
        errordlg('Please set BG first');
        return;
    end
else
    cindex = DATA.constants(constantsindex).BGi(1);
    findex = DATA.constants(constantsindex).BGi(2);
end
BG = DATA.constructs(cindex).datafile(findex).BG;
cfpb = cfp - BG(1);
yfpb = yfp - BG(2);
fretb = fret - BG(3);
BL2b = BL2 - BG(4);
BL3b = BL3 - BG(5);

%%%%%%%%%%%%%%%%%%%%%%
%   line(cfpb, fretb,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
line(cfpb, yfpb, 'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
% bin, then fit binned data
xbin = linspace(0,1e4,50);
n = 1;
for m=1:length(xbin)-1
    indices = find(cfpb>xbin(m) & cfpb<=xbin(m+1));
    if length(indices)>2
        bmeanx(n) = median(cfpb(indices));
        bmeany1(n) = median(fretb(indices));
        bmeany2(n) = median(yfpb(indices));
        n = n+1;
    end
end
line(bmeanx, bmeany1,'linestyle','none','Marker','.','color','b','MarkerSize',15,'Tag','data');
line(bmeanx, bmeany2,'linestyle','none','Marker','.','color','b','MarkerSize',15,'Tag','data');
thres = 1e4;
xxx3 = linspace(0.1,thres);
p4 = polyfit(bmeanx,bmeany1,1);
f4 = polyval(p4,xxx3);
p5 = polyfit(bmeanx,bmeany2,1);
f5 = polyval(p5,xxx3);
line(xxx3,f4,'color','r','linewidth',2,'tag','data');
line(xxx3,f5,'color','r','linewidth',2,'tag','data');
xlabel('S_C_F_P','FontName','Arial');ylabel('S_F_R_E_T or S_Y_F_P','FontName','Arial');
set(gca,'xlim',[0 thres],'ylim',[0 max(f4)*1.1],'FontName','Arial');
%   axes(handles.axes4);
%   line(f5+BG(2),xxx3+BG(1),'color','r','tag','data');
%   axes(handles.axes6);

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RC1')
    RC1 = [];
else
    RC1 = DATA.constructs(conindex).datafile(fileindex).RC1;
end
if isempty(RC1)
    DATA.constructs(conindex).datafile(fileindex).RC1 = p4;
    DATA.constructs(conindex).datafile(fileindex).RC1thres = thres;
    set(handles.RC1_edit,'string',num2str(p4(1)));
    RC1 = p4;
else
    set(handles.RC1_edit,'string',num2str(RC1(1)));
    f = polyval(RC1,xxx3);
    line(xxx3,f,'color','m','linewidth',2,'tag','data');
end

text(0.1, 0.8, ['RC1 = ' num2str(p4(1))],'Units','Normalized');

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RC2')
    RC2 = [];
else
    RC2 = DATA.constructs(conindex).datafile(fileindex).RC2;
end
if isempty(RC2)
    DATA.constructs(conindex).datafile(fileindex).RC2 = p5;
    set(handles.RC2_edit,'string',num2str(p5(1)));
    RC2 = p5;
else
    set(handles.RC2_edit,'string',num2str(RC2(1)));
    f = polyval(RC2,xxx3);
    line(xxx3,f,'color','m','linewidth',2,'tag','data');
end
```

```
                text(0.1, 0.7, ['RC2 = ' num2str(p5(1))],'Units','Normalized');

                axes(handles.axes7);
%                 linkaxes([handles.axes6,handles.axes7],'x');
                linkaxes([handles.axes6,handles.axes7],'off');
                xi = linspace(-500,500,100);
                f = ksdensity((yfpb),xi);
                line(xi,f,'tag','data','color',color); set(gca,'XScale','lin','YScale','lin');
                xlabel('S_Y_F_P');xlim([-500 500]); ylim('auto');

                %%%%%%%%%
                % bin, then fit binned data
                ybin = linspace(0,5e4,50);
                n = 1;
                bmeanx = []; bmeany = [];
                for m=1:length(ybin)-1
                    indices = find(BL3b>=ybin(m) & BL3b<=ybin(m+1));
                    if length(indices)>2
                        bmeany(n) = median(BL3b(indices));
                        bmeanx(n) = median(yfpb(indices));
                        n = n+1;
                    end
                end
                axes(handles.axes8);
                line(yfpb, BL3b,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
%                 line(fretb,BL3b,'linestyle','none','Marker','.','color','b','MarkerSize',3,'Tag','data');
                line(bmeanx, bmeany,'linestyle','none','Marker','.','color','b','MarkerSize',15,'Tag','data');
                thres = 3000;
                xxx3 = linspace(0,1500);
                p4 = polyfit(bmeanx,bmeany,1);
                f4 = polyval(p4,xxx3);
                line(xxx3,f4,'color','r','linewidth',2,'tag','data');
                xlabel('S_Y_F_P','FontName','Arial');ylabel('BL3','FontName','Arial');
                set(gca,'xlim',[0 1500],'ylim',[0 1e5],'FontName','Arial');

                if ~isfield(DATA.constructs(conindex).datafile(fileindex),'RC3')
                    RC3 = [];
                else
                    RC3 = DATA.constructs(conindex).datafile(fileindex).RC3;
                end
                if isempty(RC3)
                    DATA.constructs(conindex).datafile(fileindex).RC3 = p4;
%                     DATA.constructs(conindex).datafile(fileindex).RC3thres = thres;
                    set(handles.RC3_edit,'string',num2str(p4(1)));
                    RC3 = p4;
                else
                    set(handles.RC3_edit,'string',num2str(RC3(1)));
                    f = polyval(RC3,xxx3);
                    line(xxx3,f,'color','m','linewidth',2,'tag','data');
                end

                text(0.1, 0.8, ['RC3 = ' num2str(p4(1))],'Units','Normalized');

                axes(handles.axes9);
                linkaxes([handles.axes8,handles.axes9],'x');

                fity4 = polyval(p4,yfpb);
                errory = (fity4-BL3b)./fity4;
                line(yfpb,errory,'linestyle','none','Marker','.','color',color,'MarkerSize',3,'Tag','data');
                ylim([-1 1]);
                line(get(gca,'xlim'),[0,0],'Linestyle',':','color','r','tag','data');



%%
            case 'FR'
                moveon = 1;
                % get constants for this date
                if ~isfield(DATA.constants,'BGi') | ~isfield(DATA.constants,'RAi') | ~isfield(DATA.constants,'RDi') |
~isfield(DATA.constants,'MAMDi')
                    errordlg('Please set constants first');
                    moveon = 0;
                end
                if isempty(DATA.constants(constantsindex).BGi) | isempty(DATA.constants(constantsindex).RAi) ...
                        | isempty(DATA.constants(constantsindex).RDi) | isempty(DATA.constants(constantsindex).MAMDi)
                    errordlg('Please set constants first');
                    moveon = 0;
                end
                if moveon
                    ccindex = DATA.constants(constantsindex).BGi(1);
                    cfindex = DATA.constants(constantsindex).BGi(2);

                    BG = DATA.constructs(ccindex).datafile(cfindex).BG;

                    ccindex = DATA.constants(constantsindex).RAi(1);
                    cfindex = DATA.constants(constantsindex).RAi(2);

                    RAl = DATA.constructs(ccindex).datafile(cfindex).RAl;
%                     RAh = DATA.constructs(ccindex).datafile(cfindex).RAh;
                    RAthres = DATA.constructs(ccindex).datafile(cfindex).RAthres;
                    RA2 = DATA.constructs(ccindex).datafile(cfindex).RA2;
```

```
                set(handles.RA_edit,'string',num2str(RA1(1)));
                try
                    set(handles.RA2_edit,'string',num2str(RA2(1)));
                end

                ccindex = DATA.constants(constantsindex).RDi(1);
                cfindex = DATA.constants(constantsindex).RDi(2);

                RD1l = DATA.constructs(ccindex).datafile(cfindex).RD1l;
%                 RD1h = DATA.constructs(ccindex).datafile(cfindex).RD1h;
                RD1thres = DATA.constructs(ccindex).datafile(cfindex).RD1thres;
                set(handles.RD1_edit,'string',num2str(RD1l(1)));

                RD2l = DATA.constructs(ccindex).datafile(cfindex).RD2l;
%                 RD2h = DATA.constructs(ccindex).datafile(cfindex).RD2h;
                set(handles.RD2_edit,'string',num2str(RD2l(1)));

                RD3 = DATA.constructs(ccindex).datafile(cfindex).RD3;
                try
                    set(handles.RD3_edit,'string',num2str(RD3(1)));
                end
                if get(handles.cherrycomp,'Value')
                    ccindex = DATA.constants(constantsindex).RCi(1);
                    cfindex = DATA.constants(constantsindex).RCi(2);

                    RC1 = DATA.constructs(ccindex).datafile(cfindex).RC1;
                    RC2 = DATA.constructs(ccindex).datafile(cfindex).RC2;
                    RC3 = DATA.constructs(ccindex).datafile(cfindex).RC3;
                end

                ccindex = DATA.constants(constantsindex).MAMDi(1);
                cfindex = DATA.constants(constantsindex).MAMDi(2);

                GAGD = DATA.constructs(ccindex).datafile(cfindex).GAGD;
                FAFD = DATA.constructs(ccindex).datafile(cfindex).FAFD;

                set(handles.GAGD_edit,'string',num2str(GAGD));
                set(handles.FAFD_edit,'string',num2str(FAFD));

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 GAFA = RA1(end-1)/0.2689;
                GAFA = RA1(end-1)/0.307662905;
                GA = 0.1; FA = GAFA/GA;
                GD = GA/GAGD; FD = FA/FAFD;
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                markersize = 2;

                cfp = handles.cfp(find(filtered1));
                yfp = handles.yfp(find(filtered1));
                fret = handles.fret(find(filtered1));
                BL3 = handles.BL3(find(filtered1));

                cfp = cfp - BG(1);
                yfp = yfp - BG(2);
                fret = fret - BG(3);
                BL3 = BL3 - BG(5);

                if ~strcmpi(get(handles.showExp,'Checked'),'on')
                    % plot RD1 fit on axes5, RA fit on axes4
                    RD1x1 = logspace(log10(min(cfp)),log10(RD1thres),100);
%                     RD1x2 = logspace(log10(RD1thres),log10(max(cfp)),50);
                    RD1y1 = polyval(RD1l,RD1x1);
%                     RD1y2 = polyval(RD1h,RD1x2);
%                     RD1x = [RD1x1, RD1x2] + BG(1);
%                     RD1y = [RD1y1, RD1y2] + BG(3);

                    handles.axes5xlabel = 'S_C_F_P';
                    handles.axes5ylabel = 'S_F_R_E_T';
                    axes(handles.axes5);
                    line(RD1x1,RD1y1,'linewidth',2,'color',color,'tag','data');
                end

                % get binding constants
                if strcmp(get(gcbo,'tag'),'ErrVSKD')
                    KD = str2num(get(handles.KD,'string'));
                else
                    if ~isfield(DATA.constructs(conindex).datafile(fileindex),'KD')
                        DATA.constructs(conindex).datafile(fileindex).KD = [];
                    end
                    if ~isempty(DATA.constructs(conindex).datafile(fileindex).KD)
                        KD = DATA.constructs(conindex).datafile(fileindex).KD;
                    else
                        KD = str2num(get(handles.KD,'string'));
                        if isempty(KD)
                            KD = 3500;
                        end
                        DATA.constructs(conindex).datafile(fileindex).KD = KD;
                    end
                end
                if ~isfield(DATA.constructs(conindex).datafile(fileindex),'EFRmax')
                    DATA.constructs(conindex).datafile(fileindex).EFRmax = [];
```

```
end
if strcmp(get(gcbo,'tag'),'ErrVSKD')
    EFRmax = str2num(get(handles.EFRmax,'string'));
else
    if ~isempty(DATA.constructs(conindex).datafile(fileindex).EFRmax)
        EFRmax = DATA.constructs(conindex).datafile(fileindex).EFRmax;
    else
        EFRmax = str2num(get(handles.EFRmax,'string'));
        if isempty(EFRmax)
            EFRmax = 0.3;
        end
        DATA.constructs(conindex).datafile(fileindex).EFRmax = EFRmax;
    end
end

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'spurFR')
    DATA.constructs(conindex).datafile(fileindex).spurFR = [];
end
if ~isempty(DATA.constructs(conindex).datafile(fileindex).spurFR)
    spurFR = DATA.constructs(conindex).datafile(fileindex).spurFR;
else
    spurFR = str2num(get(handles.spurFR,'string'));
    if isempty(spurFR)
        spurFR = 5e-7;
    end
    DATA.constructs(conindex).datafile(fileindex).spurFR = spurFR;
end

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'KD2')
    DATA.constructs(conindex).datafile(fileindex).KD2 = [];
end
if ~isempty(DATA.constructs(conindex).datafile(fileindex).KD2)
    KD2 = DATA.constructs(conindex).datafile(fileindex).KD2;
else
    KD2 = str2num(get(handles.KD2,'string'));
    if isempty(KD2)
        KD2 = 3500;
    end
    DATA.constructs(conindex).datafile(fileindex).KD2 = KD2;
end

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'EFRETmax')
    DATA.constructs(conindex).datafile(fileindex).EFRETmax = [];
end
if ~isempty(DATA.constructs(conindex).datafile(fileindex).EFRETmax)
    EFRETmax = DATA.constructs(conindex).datafile(fileindex).EFRETmax;
else
    EFRETmax = str2num(get(handles.EFRETmax,'string'));
    if isempty(EFRETmax)
        EFRETmax = 0.3;
    end
    DATA.constructs(conindex).datafile(fileindex).EFRETmax = EFRETmax;
end

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'spurEFRET')
    DATA.constructs(conindex).datafile(fileindex).spurEFRET = [];
end
if ~isempty(DATA.constructs(conindex).datafile(fileindex).spurEFRET)
    spurEFRET = DATA.constructs(conindex).datafile(fileindex).spurEFRET;
else
    spurEFRET = str2num(get(handles.spurEFRET,'string'));
    if isempty(spurEFRET)
        spurEFRET = 1e-7;
    end
    DATA.constructs(conindex).datafile(fileindex).spurEFRET = spurEFRET;
end


set(handles.KD,'string',num2str(KD));
set(handles.EFRmax,'string',num2str(EFRmax));
set(handles.spurFR,'string',num2str(spurFR));
set(handles.KD2,'string',num2str(KD2));
set(handles.EFRETmax,'string',num2str(EFRETmax));
set(handles.spurEFRET,'string',num2str(spurEFRET));


% plot FR
underyfpthres = (yfp<=RAthres);
overyfpthres = (yfp>RAthres);

undercfpthres = (cfp<=RD1thres);
overcfpthres = (cfp>RD1thres);

ind = find(underyfpthres & undercfpthres);
RA = RA1; RD1 = RD11; RD2 = RD21;
if get(handles.cherrycomp,'Value')
    params = [RD1(1),RD2(1),RD3(1),RA(1),RA2(1),RC1(1),RC2(1),RC3(1)];
    [CFPd, YFPd, YFPf, Cherry] = Fdecomp(params,cfp(ind),yfp(ind),fret(ind),BL3(ind));
else
%       CFPd = polyval(RD1,cfp(ind));
%       YFPd = polyval(RA,(yfp(ind) - polyval(RD2,cfp(ind))));
```

```
                        CFPd = RD1(1)*cfp(ind);
                        YFPd = RA(1)*(yfp(ind) - RD2(1)*cfp(ind));
                        YFPf = fret(ind) - CFPd - YFPd;
                    end

                    Eeff1 = GAGD*(YFPf./YFPd) - spurFR*cfp(ind);
                    YFPest1 = YFPd/(FA*GA);
                    CFPest1 = CFPd/(FD*GD) + Eeff1.*YFPest1;
%

                    Eeff = Eeff1; YFPest = YFPest1; CFPest = CFPest1;
                    if strcmp(get(gcbo,'tag'),'ErrVSKD')
                        choice = inputdlg('Enter range of KD''s to try','ErrVSKD',1,{'logspace(0,6,50)'});
                        KDs = eval(choice{1});
                        choice = inputdlg('Enter range of Emax''s to try','ErrVSKD',1,{'linspace(0.1,0.8,50)'});
                        Emaxs = eval(choice{1});
                        for k=1:length(KDs)
                            for m=1:length(Emaxs)
                                Ab1 = (CFPest+YFPest+KDs(k) - sqrt((CFPest+YFPest+KDs(k)).^2 -
4*CFPest.*YFPest))./(2*YFPest);
                                Ab2 = Eeff./Emaxs(m);
                                ii = find(Ab1>0 & Ab1<1.5 & Ab2>0 & Ab2<1.5);
                                sqerr(m,k) = sum((Ab1(ii)-Ab2(ii)).^2);
                                sstot1 = sum((Ab1(ii)-mean(Ab1(ii))).^2);
                                sstot2 = sum((Ab2(ii)-mean(Ab2(ii))).^2);
                                Rsqr1(m,k) = 1 - sqerr(m,k)/sstot1;
                                Rsqr2(m,k) = 1 - sqerr(m,k)/sstot2;
                            end
                        end
                        [XG,YG] = meshgrid(KDs,Emaxs);
                        NN = 500;
                        [XGi,YGi] = meshgrid(logspace(log10(KDs(1)),log10(KDs(end)),NN), ...
                            linspace(Emaxs(1),Emaxs(end),NN));
                        ind = find(Rsqr1>0);
                        figure;
                        newZ = interp2(XG,YG,sqerr,XGi,YGi);
%                         surf(log10(XGi),YGi,newZ,'linestyle','none');
                        contourf(XGi,YGi,log10(newZ),20);
                        set(gca,'XScale','log');
                    end
                    if handles.tryfitFR1
                        [params] = fminsearch(@(p) bindingmodel(p,CFPest,YFPest,Eeff),[KD, EFRmax],
optimset('Display','final','TolX',1e-6,'TolFun',1e-6));
                        KD = params(1);
                        EFRmax = params(2);
                        if KD<0 | EFRmax<0
                            errordlg('Bad fit.. try new params');
                            return;
                        end
                    end
                    if handles.tryfitFR2
                        [params] = fminsearch(@(p) bindingmodel2(p,CFPest,YFPest,Eeff),[KD],
optimset('Display','final','TolX',1e-6,'TolFun',1e-6));
                        KD = params(1);
                        if KD<0
                            errordlg('Bad fit.. try new params');
                            return;
                        end
                    end
                    DATA.constructs(conindex).datafile(fileindex).KD = KD;
                    DATA.constructs(conindex).datafile(fileindex).EFRmax = EFRmax;

                    set(handles.KD,'string',num2str(KD));
                    set(handles.EFRmax,'string',num2str(EFRmax));

                    if strcmpi(get(handles.R2C2adjust,'Checked'),'on')

                        syms a b c; % a = Dfree, b = Afree, c = CY
                        for Z=1:length(CFPest)
                            a = 16;
                            b = -16*(CFPest(Z)+YFPest(Z));
                            c = (4*CFPest(Z)^2+4*YFPest(Z)^2+16*CFPest(Z)*YFPest(Z));
                            d = -4*CFPest(Z)*YFPest(Z)^2 - 4*YFPest(Z)*CFPest(Z)^2 - KD;
                            e = CFPest(Z)^2*YFPest(Z)^2;
                            solns(:,Z) = roots([a,b,c,d,e]);
                        end
                        solnN = 4;
                        R2C2 = (solns(solnN,:)');
                        Dfree = CFPest - 2*R2C2;
                        Afree = YFPest - 2*R2C2;

                        Ab = R2C2./YFPest; %solns(4) = R2C2
                        Eeffpred = Ab*EFRmax;
                        EFRpred = EFRmax*(2*Dfree.^2.*Afree)./(KD + 2*Dfree.^2.*Afree);

                        axes(handles.axes6);
%                         line(Dfree,
Eeff/EFRmax,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        line(Dfree.^2.*Afree,
Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        xlabel('Dfree^2*Afree'); ylabel('Eeff');
```

```
                    else
                        Ab = (CFPest+YFPest+KD - sqrt((CFPest+YFPest+KD).^2 - 4*CFPest.*YFPest))./(2*YFPest);
                        Dfree = CFPest - Ab.*YFPest;
                        Eeffpred = Ab*EFRmax;
                        EFRpred = EFRmax*Dfree./(Dfree + KD);

                        axes(handles.axes6);
    %                     line(Dfree,
Eeff/EFRmax,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        line(Dfree,
Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        xlabel('Dfree'); ylabel('Eeff');

                    end

                    if strcmpi(get(handles.showExp,'Checked'),'on')
                        axes(handles.axes4);
                        xi = linspace(0.1,6,100);
                        f = ksdensity(log10(CFPest),xi);
                        line(xi,f,'tag','data','color',color); set(gca,'XScale','lin','YScale','lin');
                        xlabel('CFPest');
%                         hist(log10(CFPest)); xlabel('CFPest'); xlim([1 6]);
                        axes(handles.axes5);
                        f = ksdensity(log10(YFPest),xi);
                        line(xi,f,'tag','data','color',color); set(gca,'XScale','lin','YScale','lin');
                        xlabel('YFPest');
%                         hist(log10(YFPest)); xlabel('YFPest'); xlim([1 6]);

                    end
                    axes(handles.axes7);
                    if strcmpi(get(handles.R2C2adjust,'Checked'),'off')
                        line(Ab,
Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        xlabel('Ab'); ylabel('Eeff');
                        set(gca,'XScale','lin');
                    else

line(Eeff,EFRpred,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                        line([0,1],[0,1],'color','r','Tag','data');
                        xlabel('Eeff'); ylabel('EFRpred');
                        set(gca,'XScale','lin');
                    end


                    % error prediction
                    sqerr = (EFRpred-Eeff).^2;

                    if strcmp(get(gcbo,'tag'),'ErrVSKD')
                        set(handles.numinroi_edit,'string',num2str(sum(sqerr)));
                    end
                    eps = 1e-6;
                    dFRdKD = ((EFRmax*Dfree./(Dfree + (1+eps)*KD)) - EFRpred)/(eps*KD);
                    dFRdEFRmax = (((1+eps)*EFRmax*Dfree./(Dfree + KD)) - EFRpred)/(eps*EFRmax);
                    P = [ sum(dFRdKD.^2)               sum(dFRdKD.*dFRdEFRmax); ...
                          sum(dFRdKD.*dFRdEFRmax)      sum(dFRdEFRmax.^2)];
                    invP = inv(P);
                    KDstd = sqrt(invP(1,1))*sqrt(sum(sqerr)/(length(sqerr)-2));
                    EFRmaxstd = sqrt(invP(2,2))*sqrt(sum(sqerr)/(length(sqerr)-2));

                    DATA.constructs(conindex).datafile(fileindex).KDstd = KDstd;
                    DATA.constructs(conindex).datafile(fileindex).EFRmaxstd = EFRmaxstd;
                    DATA.constructs(conindex).datafile(fileindex).FRnumcells = length(sqerr);


                    if EFRmax<0.1
                        ylims = [-0.1 0.35];
                    else
                        ylims = [-0.1*EFRmax EFRmax*1.3];
                    end
                    if strcmpi(get(handles.R2C2adjust,'Checked'),'off')
                        Dfreefit = logspace(log10(KD)-2.5,log10(KD)+2.5,100);
                        FRfit = EFRmax*Dfreefit./(Dfreefit + KD);
                        axes(handles.axes6);
%                         line(Dfreefit, FRfit/EFRmax,'color','r','LineWidth',1,'Tag','data');
                        line(Dfreefit, FRfit,'color','r','LineWidth',1,'Tag','data');
                        set(handles.axes6,'XLim',[KD*0.01 KD*100],'ylim',ylims,'xscale','log');%
                    else
                        Dfreefit = logspace(log10(KD)-2.5,log10(KD)+2.5,100);
                        FRfit = EFRmax*Dfreefit./(Dfreefit + KD/2);
                        axes(handles.axes6);
                        line(Dfreefit, FRfit,'color','r','LineWidth',1,'Tag','data');
                        set(handles.axes6,'XLim',[KD*0.001 KD*1000],'ylim',ylims,'xscale','log');
                    end
                    Eefffit = [0 EFRmax];
                    Abfit = [0 1];

                    axes(handles.axes7);
                    if strcmpi(get(handles.R2C2adjust,'Checked'),'off')
                        line(Abfit,Eefffit,'color','r','LineWidth',1,'Tag','data');
```

```
                    set(handles.axes7,'xlim',[0 max(real(Ab))*1.1],'ylim',[max(min(Eeff)*1.3,-0.1)
min(max(Eeff)*1.1,1)]);
                    text(0.1, 0.9, {['KDstd = ' num2str(KDstd)]; ['EFRmaxstd = '
num2str(EFRmaxstd)]},'tag','data','Units','Normalized');
                end



                % plot EFRET
                cfp = handles.cfp(filtered2)-BG(1);
                yfp = handles.yfp(filtered2)-BG(2);
                fret = handles.fret(filtered2)-BG(3);
                BL3 = handles.BL3(find(filtered2))-BG(5);


                underyfpthres = (yfp<=RAthres);
                overyfpthres = (yfp>RAthres);

                undercfpthres = (cfp<=RD1thres);
                overcfpthres = (cfp>RD1thres);

                ind = find(underyfpthres & undercfpthres);
                RA = RA1; RD1 = RD11; RD2 = RD21;
                if get(handles.cherrycomp,'Value')
                    params = [RD1(1),RD2(1),RD3(1),RA(1),RA2(1),RC1(1),RC2(1),RC3(1)];
                    [CFPd, YFPd, YFPf, Cherry] = Fdecomp(params,cfp(ind),yfp(ind),fret(ind),BL3(ind));
                else
%                     CFPd = polyval(RD1,cfp(ind));
%                     YFPd = polyval(RA,(yfp(ind) - polyval(RD2,cfp(ind))));
                    CFPd = RD1(1)*cfp(ind);
                    YFPd = RA(1)*(yfp(ind) - RD2(1)*cfp(ind));
                    YFPf = fret(ind) - CFPd - YFPd;
                end


                EFRETeff1 = YFPf./(YFPf + CFPd*FA/FD) - spurEFRET*yfp(ind);
                YFPest1 = YFPd/(FA*GA);
                CFPest1 = CFPd./(FD*GD*(1-EFRETeff1));

                EFRETeff = EFRETeff1; YFPest = YFPest1; CFPest = CFPest1;

                if handles.tryfitEFRET1
                    [params] = fminsearch(@(p) bindingmodel3(p,CFPest,YFPest,EFRETeff),[KD2, EFRETmax],
optimset('Display','final','TolX',1e-6,'TolFun',1e-6));
                    KD2 = params(1);
                    EFRETmax = params(2);
                    if KD2<0 | EFRETmax<0
                        errordlg('Bad fit.. try new params');
                        return;
                    end
                end
                if handles.tryfitEFRET2
                    [params] = fminsearch(@(p) bindingmodel4(p,CFPest,YFPest,EFRETeff),[KD2],
optimset('Display','final','TolX',1e-6,'TolFun',1e-6));
                    KD2 = params(1);
                    if KD2<0
                        errordlg('Bad fit.. try new params');
                        return;
                    end
                end
                DATA.constructs(conindex).datafile(fileindex).KD2 = KD2;
                DATA.constructs(conindex).datafile(fileindex).EFRETmax = EFRETmax;

                set(handles.KD2,'string',num2str(KD2));
                set(handles.EFRETmax,'string',num2str(EFRETmax));

                Db = (CFPest+YFPest+KD2 - sqrt((CFPest+YFPest+KD2).^2 - 4*CFPest.*YFPest))./(2*CFPest);
                todel = find(imag(Db));
%                   if length(todel)>5
%                       warndlg([num2str(length(todel)) ' imaginary Db''s detected']);
%                   end
                Db(todel) = [];
                CFPest(todel) = [];
                YFPest(todel) = [];
                EFRETeff(todel) = [];

                Afree = YFPest - Db.*CFPest;

                EFRETeffpred = Db*EFRETmax;
                EFRETpred = EFRETmax*Afree./(Afree + KD2);

                axes(handles.axes8);
                line(Afree,
EFRETeff,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                axes(handles.axes9);

                line(Db,
EFRETeff,'linestyle','none','Marker','.','color',color,'MarkerSize',markersize,'Tag','data');
                xlabel('Db'); ylabel('EFRETeff');
                set(gca,'XScale','lin');
```

```
                    % error prediction
                    sqerr = (EFRETpred-EFRETeff).^2;
                    eps = 1e-6;
                    dEFRETdKD = ((EFRETmax*Afree./(Afree + (1+eps)*KD2)) - EFRETpred)/(eps*KD2);
                    dEFRETdEFRETmax = (((1+eps)*EFRETmax*Afree./(Afree + KD2)) - EFRETpred)/(eps*EFRETmax);
                    P = [ sum(dEFRETdKD.^2)               sum(dEFRETdKD.*dEFRETdEFRETmax); ...
                          sum(dEFRETdKD.*dEFRETdEFRETmax)      sum(dEFRETdEFRETmax.^2)];
                    invP = inv(P);
                    KD2std = sqrt(invP(1,1))*sqrt(sum(sqerr)/(length(sqerr)-2));
                    EFRETmaxstd = sqrt(invP(2,2))*sqrt(sum(sqerr)/(length(sqerr)-2));

                    DATA.constructs(conindex).datafile(fileindex).KD2std = KD2std;
                    DATA.constructs(conindex).datafile(fileindex).EFRETmaxstd = EFRETmaxstd;
                    DATA.constructs(conindex).datafile(fileindex).EFRETnumcells = length(sqerr);

                    Afreefit = logspace(log10(KD2)-2.5,log10(KD2)+2.5,1000);
                    EFRETfit = EFRETmax*Afreefit./(Afreefit + KD2);
                    EFRETefffit = [0 EFRETmax];
                    Dbfit = [0 1];
                    axes(handles.axes8);
                    line(Afreefit, EFRETfit,'color','r','LineWidth',1,'Tag','data');

                    axes(handles.axes9);
%                     if strcmpi(get(handles.showExp,'Checked'),'off')
                        line(Dbfit,EFRETefffit,'color','r','LineWidth',1,'Tag','data');
                        set(handles.axes9,'xlim',[0 max(Db)*1.1],'ylim',[max(min(EFRETeff)*1.3,-0.1)
min(max(EFRETeff)*1.1,1)]);
                        text(0.1, 0.9, {['KD2std = ' num2str(KD2std)]; ['EFRETmaxstd = '
num2str(EFRETmaxstd)]},'tag','data','Units','Normalized');
%                     end
                    if EFRETmax<0.1
                        ylims = [-0.1 0.35];
                    else
                        ylims = [-0.1*EFRETmax EFRETmax*1.3];
                    end
                    set(handles.axes8,'XLim',[KD2*0.01 KD2*100],'ylim',ylims,'xscale','log');

                end

%                 binding_Callback(handles.binding, [], handles);
%%
            case 'simple'
                moveon = 1;
                % get constants for this date
                if ~isfield(DATA.constants,'BGi') | ~isfield(DATA.constants,'RAi') | ~isfield(DATA.constants,'RDi') |
~isfield(DATA.constants,'MAMDi')
                    warndlg('Please set constants first');
                    moveon = 0;
                end
                if isempty(DATA.constants(constantsindex).BGi) | isempty(DATA.constants(constantsindex).RAi) ...
                        | isempty(DATA.constants(constantsindex).RDi) | isempty(DATA.constants(constantsindex).MAMDi)
                    warndlg('Please set constants first');
                    moveon = 0;
                end

                if moveon
                    ccindex = DATA.constants(constantsindex).BGi(1);
                    cfindex = DATA.constants(constantsindex).BGi(2);

                    BG = DATA.constructs(ccindex).datafile(cfindex).BG;

                    ccindex = DATA.constants(constantsindex).RAi(1);
                    cfindex = DATA.constants(constantsindex).RAi(2);

                    RAl = DATA.constructs(ccindex).datafile(cfindex).RAl;
%                     RAh = DATA.constructs(ccindex).datafile(cfindex).RAh;
                    RAthres = DATA.constructs(ccindex).datafile(cfindex).RAthres;
                    RA2 = DATA.constructs(ccindex).datafile(cfindex).RA2;
                    set(handles.RA_edit,'string',num2str(RAl(1)));
                    try
                        set(handles.RA2_edit,'string',num2str(RA2(1)));
                    end

                    ccindex = DATA.constants(constantsindex).RDi(1);
                    cfindex = DATA.constants(constantsindex).RDi(2);

                    RD1l = DATA.constructs(ccindex).datafile(cfindex).RD1l;
%                     RD1h = DATA.constructs(ccindex).datafile(cfindex).RD1h;
                    RD1thres = DATA.constructs(ccindex).datafile(cfindex).RD1thres;
                    set(handles.RD1_edit,'string',num2str(RD1l(1)));

                    RD2l = DATA.constructs(ccindex).datafile(cfindex).RD2l;
%                     RD2h = DATA.constructs(ccindex).datafile(cfindex).RD2h;
                    set(handles.RD2_edit,'string',num2str(RD2l(1)));

                    RD3 = DATA.constructs(ccindex).datafile(cfindex).RD3;
                    try
                        set(handles.RD3_edit,'string',num2str(RD3(1)));
                    end
```

```
if get(handles.cherrycomp,'Value')
    ccindex = DATA.constants(constantsindex).RCi(1);
    cfindex = DATA.constants(constantsindex).RCi(2);

    RC1 = DATA.constructs(ccindex).datafile(cfindex).RC1;
    RC2 = DATA.constructs(ccindex).datafile(cfindex).RC2;
    RC3 = DATA.constructs(ccindex).datafile(cfindex).RC3;
end

ccindex = DATA.constants(constantsindex).MAMDi(1);
cfindex = DATA.constants(constantsindex).MAMDi(2);

GAGD = DATA.constructs(ccindex).datafile(cfindex).GAGD;
FAFD = DATA.constructs(ccindex).datafile(cfindex).FAFD;

set(handles.GAGD_edit,'string',num2str(GAGD));
set(handles.FAFD_edit,'string',num2str(FAFD));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   GAFA = RAl(end-1)/0.2689;
GAFA = RAl(end-1)/0.307662905;
GA = 1; FA = GAFA/GA;
GD = GA/GAGD; FD = FA/FAFD;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

cfp = handles.cfp(find(filtered1));
yfp = handles.yfp(find(filtered1));
fret = handles.fret(find(filtered1));
BL3 = handles.BL3(find(filtered1));
flowtime = handles.flowtime(find(filtered1));

cfp = cfp - BG(1);
yfp = yfp - BG(2);
fret = fret - BG(3);
BL3 = BL3 - BG(5);

% plot RD1 fit on axes5
RD1x1 = logspace(log10(min(cfp)),log10(RD1thres),100);
%   RD1x2 = logspace(log10(RD1thres),log10(max(cfp)),50);
RD1y1 = polyval(RD1l,RD1x1);
%   RD1y2 = polyval(RD1h,RD1x2);
%   RD1x = [RD1x1, RD1x2] + BG(1);
%   RD1y = [RD1y1, RD1y2] + BG(3);

handles.axes5xlabel = 'S_C_F_P';
handles.axes5ylabel = 'S_F_R_E_T';
axes(handles.axes5);
line(RD1x1,RD1y1,'linewidth',2,'color',color,'tag','data');

if ~isfield(DATA.constructs(conindex).datafile(fileindex),'spurFR')
    DATA.constructs(conindex).datafile(fileindex).spurFR = [];
end
if ~isempty(DATA.constructs(conindex).datafile(fileindex).spurFR)
    spurFR = DATA.constructs(conindex).datafile(fileindex).spurFR;
else
    spurFR = str2num(get(handles.spurFR,'string'));
    if isempty(spurFR)
        spurFR = 5e-7;
    end
    DATA.constructs(conindex).datafile(fileindex).spurFR = spurFR;
end
set(handles.spurFR,'string',spurFR);

underyfpthres = (yfp<=RAthres);
overyfpthres = (yfp>RAthres);

undercfpthres = (cfp<=RD1thres);
overcfpthres = (cfp>RD1thres);

ind = find(underyfpthres & undercfpthres);

RA = RAl; RD1 = RD1l; RD2 = RD2l;
if get(handles.cherrycomp,'Value')
    params = [RD1(1),RD2(1),RD3(1),RA(1),RA2(1),RC1(1),RC2(1),RC3(1)];
    [CFPd, YFPd, YFPf, Cherry] = Fdecomp(params,cfp(ind),yfp(ind),fret(ind),BL3(ind));
else
%       CFPd = polyval(RD1,cfp(ind));
%       YFPd = polyval(RA,(yfp(ind) - polyval(RD2,cfp(ind))));
    CFPd = RD1(1)*cfp(ind);
    YFPd = RA(1)*(yfp(ind) - RD2(1)*cfp(ind));
    YFPf = fret(ind) - CFPd - YFPd;
end
flowtime = flowtime(ind);
flowtime = flowtime/60000;
Eeff = GAGD*(YFPf./YFPd) - spurFR*cfp(ind);

YFPest1 = YFPd/(FA*GA);
CFPest1 = CFPd/(FD*GD) + Eeff.*YFPest1;

CFPest = CFPest1; YFPest = YFPest1;
```

```
                    if strcmp(get(handles.plotvsred,'Checked'),'on')
                        axes(handles.axes6);
                        line(Cherry, Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        xlabel('Red'); ylabel('Eeff');
                        xlim([1 1e5]); ylim([-0.1 0.7]);
                        axes(handles.axes7);
                        line(Cherry,
YFPest./CFPest,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        ylim([-0.5 2]); xlim([1 1e5]);
                        line([1 1e6],[1 1],'linestyle',':','color','r','linewidth',2,'tag','data');
                        ylabel('NA/ND'); xlabel('Red');
                        meanNAND = trimmean(YFPest./CFPest,0.1);
                        text(0.1,0.9,['Mean NA/ND = ' num2str(meanNAND)],'Units','Normalized');

                    else
                        axes(handles.axes6);
                        if strcmpi(get(handles.plotvstime,'Checked'),'on')
                            line(flowtime,
Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                            xlabel('Time'); ylabel('Eeff');
                            set(handles.axes6,'xlim',[0 max(flowtime)+0.1],'ylim',[-0.1 0.7]);
                        else
                            line(CFPest,
Eeff,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                            xlabel('CFPest'); ylabel('Eeff');
                            set(handles.axes6,'xlim',[1e3 10e4],'ylim',[-0.1 0.7]);
                        end
                        line(get(gca,'xlim'),[0 0],'linestyle',':','color','r','tag','data');


                        axes(handles.axes7);

                        if strcmpi(get(handles.plotvstime,'Checked'),'on')
                            line(flowtime,
YFPest./CFPest,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                            xlabel('Time'); ylabel('NA/ND');
                            set(handles.axes7,'xlim',[0 max(flowtime)+0.1],'ylim',[-0.5 2]);
                        else
                            line(CFPest,
YFPest./CFPest,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                            ylim([-0.5 2]); xlim([1e2 1e5]);
                            line([1e2 1e6],[1 1],'linestyle',':','color','r','linewidth',2,'tag','data');
                            ylabel('NA/ND'); xlabel('CFPest');
                            meanNAND = trimmean(YFPest./CFPest,0.1);
                            text(0.1,0.9,['Mean NA/ND = ' num2str(meanNAND)],'Units','Normalized');
                        end

                    end

                    axes(handles.axes8);
                    cfp = handles.cfp(find(filtered2));
                    yfp = handles.yfp(find(filtered2));
                    fret = handles.fret(find(filtered2));
                    BL3 = handles.BL3(find(filtered2));

                    cfp = cfp - BG(1);
                    yfp = yfp - BG(2);
                    fret = fret - BG(3);
                    BL3 = BL3 - BG(5);

                    if ~isfield(DATA.constructs(conindex).datafile(fileindex),'spurEFRET')
                        DATA.constructs(conindex).datafile(fileindex).spurEFRET = [];
                    end
                    if ~isempty(DATA.constructs(conindex).datafile(fileindex).spurEFRET)
                        spurEFRET = DATA.constructs(conindex).datafile(fileindex).spurEFRET;
                    else
                        spurEFRET = str2num(get(handles.spurEFRET,'string'));
                        if isempty(spurEFRET)
                            spurEFRET = 1e-7;
                        end
                        DATA.constructs(conindex).datafile(fileindex).spurEFRET = spurEFRET;
                    end
                    set(handles.spurEFRET,'string',spurEFRET);

                    underyfpthres = (yfp<=RAthres);
                    overyfpthres = (yfp>RAthres);

                    undercfpthres = (cfp<=RD1thres);
                    overcfpthres = (cfp>RD1thres);

                    ind = find(underyfpthres & undercfpthres);
                    RA = RAl; RD1 = RD1l; RD2 = RD2l;
                    if get(handles.cherrycomp,'Value')
                        params = [RD1(1),RD2(1),RD3(1),RA(1),RA2(1),RC1(1),RC2(1),RC3(1)];
                        [CFPd, YFPd, YFPf, Cherry] = Fdecomp(params,cfp(ind),yfp(ind),fret(ind),BL3(ind));
                    else
%                         CFPd = polyval(RD1,cfp(ind));
%                         YFPd = polyval(RA,(yfp(ind) - polyval(RD2,cfp(ind))));
                        CFPd = RD1(1)*cfp(ind);
                        YFPd = RA(1)*(yfp(ind) - RD2(1)*cfp(ind));
                        YFPf = fret(ind) - CFPd - YFPd;
```

```
                    end

                    x1 = YFPd;
                    y1 = YFPf./(YFPf + CFPd*FA/FD) - spurEFRET*yfp(ind);
                    YFPest1 = YFPd/(FA*GA);
                    CFPest1 = CFPd./(FD*GD*(1-y1));

                    x = x1; y = y1; CFPest = CFPest1; YFPest = YFPest1;

                    if strcmp(get(handles.plotvsred,'Checked'),'on')
                        axes(handles.axes8);
                        line(Cherry, y,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        xlabel('Red'); ylabel('EFRETeff');
                        xlim([1 1e5]); ylim([-0.1 0.7]);
                        axes(handles.axes9);
                        line(Cherry,
YFPest./CFPest,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        ylim([-0.5 2]); xlim([1 1e5]);
                        line([1 1e6],[1 1],'linestyle',':','color','r','linewidth',2,'tag','data');
                        ylabel('NA/ND'); xlabel('Red');
                        meanNAND = trimmean(YFPest./CFPest,0.1);
                        text(0.1,0.9,['Mean NA/ND = ' num2str(meanNAND)],'Units','Normalized');

                    else
                        axes(handles.axes8);
                        line(YFPest, y,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        xlabel('YFPest'); ylabel('Eeff');
                        set(handles.axes8,'xlim',[1e3, 1e5],'ylim',[-0.1 0.7]);
                        line(get(gca,'xlim'),[0 0],'linestyle',':','color','r','tag','data');
                        axes(handles.axes9);
    %                     [f,xi] = ksdensity((log10(CFPest(find(CFPest>0)))));
    %                     line(xi,f,'color','b','tag','data');
    %                     [f,xi] = ksdensity((log10(YFPest(find(YFPest>0)))));
    %                     line(xi,f,'color','r','tag','data');
                        line(YFPest,
YFPest./CFPest,'linestyle','none','Marker','.','color',color,'MarkerSize',2,'Tag','data');
                        ylim([-0.5 2]); xlim([1e2 1e5]);
                        line([1e2 1e6],[1 1],'linestyle',':','color','r','linewidth',2,'tag','data');
                        ylabel('NA/ND'); xlabel('YFPest');
                        meanNAND = trimmean(YFPest./CFPest,0.1);
                        text(0.1,0.9,['Mean NA/ND = ' num2str(meanNAND)],'Units','Normalized');

                    end
                    % xlswrite('output.xlsx',[cfp, yfp, fret, EFRETeff],'EFRET');
                end
            end
        end
else

end


save(handles.analysisfile,'-struct','DATA','-append');



%%
% --------------------------------------------------------------------
function roi1_Callback(hObject, eventdata, handles)

handles.roi1poly = impoly(handles.axes1);
setColor(handles.roi1poly,'red');
handles.roi1pos = getPosition(handles.roi1poly);
addNewPositionCallback(handles.roi1poly,@roi1Update);
temph = get(handles.axes1,'Children');
set(handles.axes1,'Children',[temph(end); temph(1:end-1)]);
guidata(hObject,handles);
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ','''tag'',''data''))']);
end
UpdateFlowFretC;

function roi1Update(pos)
handles = guidata(gcf);
handles.roi1pos = pos;
guidata(gcf,handles);

% --------------------------------------------------------------------
function roi2_Callback(hObject, eventdata, handles)
handles.roi2poly = impoly(handles.axes2);
setColor(handles.roi2poly,[0 1 0]);
handles.roi2pos = getPosition(handles.roi2poly);
addNewPositionCallback(handles.roi2poly,@roi2Update);
temph = get(handles.axes2,'Children');
set(handles.axes2,'Children',[temph(end); temph(1:end-1)]);
guidata(hObject,handles);
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ','''tag'',''data''))']);
end
UpdateFlowFretC;
```

```
function roi2Update(pos)
handles = guidata(gcf);
handles.roi2pos = pos;
guidata(gcf,handles);

% --------------------------------------------------------------------
function roi3_Callback(hObject, eventdata, handles)
handles.roi3poly = impoly(handles.axes3);
setColor(handles.roi3poly,[1 0 1]);
handles.roi3pos = getPosition(handles.roi3poly);
addNewPositionCallback(handles.roi3poly,@roi3Update);
temph = get(handles.axes3,'Children');
set(handles.axes3,'Children',[temph(end); temph(1:end-1)]);
guidata(hObject,handles);
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;

function roi3Update(pos)
handles = guidata(gcf);
handles.roi3pos = pos;
guidata(gcf,handles);

% --------------------------------------------------------------------
function deleteroi3_Callback(hObject, eventdata, handles)
if isfield(handles,'roi3poly')
    delete(handles.roi3poly);
    handles = rmfield(handles, 'roi3poly');
    guidata(hObject,handles);
end
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;

% --------------------------------------------------------------------
function deleteroi2_Callback(hObject, eventdata, handles)
if isfield(handles,'roi2poly')
    delete(handles.roi2poly);
    handles = rmfield(handles, 'roi2poly');
    guidata(hObject,handles);
end
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;

% --------------------------------------------------------------------
function deleteroi1_Callback(hObject, eventdata, handles)
if isfield(handles,'roi1poly')
    delete(handles.roi1poly);
    handles = rmfield(handles, 'roi1poly');
    guidata(hObject,handles);
end
for k=1:3
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;

function roi4_Callback(hObject, eventdata, handles)

handles.roi4poly = impoly(handles.axes4);
setColor(handles.roi4poly,'red');
handles.roi4pos = getPosition(handles.roi4poly);
addNewPositionCallback(handles.roi4poly,@roi4Update);
temph = get(handles.axes4,'Children');
set(handles.axes4,'Children',[temph(end); temph(1:end-1)]);
guidata(hObject,handles);
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;

function roi4Update(pos)
handles = guidata(gcf);
handles.roi4pos = pos;
guidata(gcf,handles);

% --------------------------------------------------------------------
function deleteroi4_Callback(hObject, eventdata, handles)
if isfield(handles,'roi4poly')
    delete(handles.roi4poly);
    handles = rmfield(handles, 'roi4poly');
    guidata(hObject,handles);
end
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ',''tag'',''data''))']);
end
UpdateFlowFretC;
```

```
function roi5_Callback(hObject, eventdata, handles)

handles.roi5poly = impoly(handles.axes5);
setColor(handles.roi5poly,'red');
handles.roi5pos = getPosition(handles.roi5poly);
addNewPositionCallback(handles.roi5poly,@roi5Update);
temph = get(handles.axes5,'Children');
set(handles.axes5,'Children',[temph(end); temph(1:end-1)]);
guidata(hObject,handles);
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ','''tag''','''data'''))']);
end
UpdateFlowFretC;

function roi5Update(pos)
handles = guidata(gcf);
handles.roi5pos = pos;
guidata(gcf,handles);
% UpdateFlowFretC;

% --------------------------------------------------------------------
function deleteroi5_Callback(hObject, eventdata, handles)
if isfield(handles,'roi5poly')
    delete(handles.roi5poly);
    handles = rmfield(handles, 'roi5poly');
    guidata(hObject,handles);
end
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ','''tag''','''data'''))']);
end
UpdateFlowFretC;


function fighotkeys(~,event,~)
handles = guidata(gcf);
control = 0;
alt = 0;
shift = 0;

for x=1:length(event.Modifier)
    switch(event.Modifier{x})
        case 'control'
            control = 1;
        case 'alt'
            alt = 1;
        case 'shift'
            shift = 1;
    end
end
if alt & strcmp(event.Key,'return')
    listbox_Callback(handles.listbox,[],handles);
    guidata(handles.listbox,handles);
end
if ~alt & strcmp(event.Key,'return')
    UpdateFlowFretC;
end

% --------------------------------------------------------------------

%%

% --------------------------------------------------------------------
function estspur2_Callback(hObject, eventdata, handles)
% linear fit to estimate spurious FRET
set(handles.spurEFRET,'string','');
UpdateFlowFretC(1);
[x,y] = ginput(2);
xdata = get(findobj(handles.axes8,'type','line','tag','data','marker','.'),'xdata');
ydata = get(findobj(handles.axes8,'type','line','tag','data','marker','.'),'ydata');
tofit = find(xdata>x(1) & xdata<x(2));
p = polyfit(xdata(tofit)',ydata(tofit)',1); % unconstrained
xx = xlim;
yy = xx*p(1) + p(2);
line(xx,yy,'color','r');
text(mean(xx), mean(yy), {['slope = ' num2str(p(1),3)]; ['intersect = ' num2str(p(2),2)]});
set(handles.spurEFRET,'string',num2str(p(1),3));


% --------------------------------------------------------------------
function estspur1_Callback(hObject, eventdata, handles)
% linear fit to estimate spurious FRET
set(handles.spurFR,'string','');
UpdateFlowFretC(1);
[x,y] = ginput(2);
xdata = get(findobj(handles.axes6,'type','line','tag','data','marker','.'),'xdata');
ydata = get(findobj(handles.axes6,'type','line','tag','data','marker','.'),'ydata');
tofit = find(xdata>x(1) & xdata<x(2));
% p = lsqlin(xdata(tofit)',ydata(tofit)',[],[],0,0); % constrain fit to go through [0,0]
p = polyfit(xdata(tofit)',ydata(tofit)',1); % unconstrained
xx = xlim;
yy = xx*p(1) + p(2);
```

```
line(xx,yy,'color','r');
text(mean(xx), mean(yy), {['slope = ' num2str(p(1),3)]; ['intersect = ' num2str(p(2),2)]});
set(handles.spurFR,'string',num2str(p(1),2));


% --------------------------------------------------------------------
function setBG_Callback(hObject, eventdata, handles)

DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));
DATA.constructs(conindex).plottype = 'BG';
save(handles.analysisfile,'-struct','DATA','-append');
% h = waitbar(0,'Please wait... updating files (ignore the beeps!)','WindowStyle','modal');
% liststr = get(handles.listbox,'string');
% for k=1:length(liststr)
%     waitbar(k/length(liststr),h);
%     set(handles.listbox,'Value',k);
%     listbox_Callback(handles.listbox,[],handles);
%     guidata(hObject,handles);
% end
% close(h);

% --------------------------------------------------------------------
function setRA_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));
DATA.constructs(conindex).plottype = 'RA';
save(handles.analysisfile,'-struct','DATA','-append');
% h = waitbar(0,'Please wait... updating files (ignore the beeps!)','WindowStyle','modal');
% liststr = get(handles.listbox,'string');
% for k=1:length(liststr)
%     waitbar(k/length(liststr),h);
%     set(handles.listbox,'Value',k);
%     listbox_Callback(handles.listbox,[],handles);
%     guidata(hObject,handles);
% end
% close(h);

% --------------------------------------------------------------------
function setRD_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));
DATA.constructs(conindex).plottype = 'RD';
save(handles.analysisfile,'-struct','DATA','-append');
% h = waitbar(0,'Please wait... updating files (ignore the beeps!)','WindowStyle','modal');
% liststr = get(handles.listbox,'string');
% for k=1:length(liststr)
%     waitbar(k/length(liststr),h);
%     set(handles.listbox,'Value',k);
%     listbox_Callback(handles.listbox,[],handles);
%     guidata(hObject,handles);
% end
% close(h);

% --------------------------------------------------------------------
function setRC_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));
DATA.constructs(conindex).plottype = 'RC';
save(handles.analysisfile,'-struct','DATA','-append');
% h = waitbar(0,'Please wait... updating files (ignore the beeps!)','WindowStyle','modal');
% liststr = get(handles.listbox,'string');
% for k=1:length(liststr)
```

```
%     waitbar(k/length(liststr),h);
%     set(handles.listbox,'Value',k);
%     listbox_Callback(handles.listbox,[],handles);
%     guidata(hObject,handles);
% end
% close(h);

% --------------------------------------------------------------------
function simple_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
for k=1:length(listboxval)
    conname = listboxstr{listboxval(k)};
    conindex = find(strcmp({DATA.constructs.name},conname));
    DATA.constructs(conindex).plottype = 'simple';
end
save(handles.analysisfile,'-struct','DATA','-append');

function binding_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
for k=1:length(listboxval)
    conname = listboxstr{listboxval(k)};
    conindex = find(strcmp({DATA.constructs.name},conname));
    DATA.constructs(conindex).plottype = 'FR';
end
save(handles.analysisfile,'-struct','DATA','-append');



% --------------------------------------------------------------------
function deconstruct_Callback(hObject, eventdata, handles)
listboxstr = get(handles.listbox,'string');
listboxval = get(handles.listbox,'value');
DATA = load(handles.analysisfile);
sel = find(strcmp(DATA.names,listboxstr{listboxval}));
structname = ['STR', num2str(sel,'%04.0f')];
clear DATA;
DATA = load(handles.analysisfile,structname);
load(handles.analysisfile,'RA','RD1','RD2','BG','filepath');
filtered = find(DATA.(structname).filtertable(:,5)); % use filtered1
filename = [filepath, '\' DATA.(structname).filename];
[fcsdat, fcshdr] = fca_readfcs(filename);
cfpr = fcsdat(:,find(strcmpi(handles.CFPname,handles.parnames)));
yfpr = fcsdat(:,find(strcmpi(handles.YFPname,handles.parnames)));
fretr = fcsdat(:,find(strcmpi(handles.FRETname,handles.parnames)));

cfp = cfpr(filtered) - BG(1);
yfp = yfpr(filtered) - BG(2);
fret = fretr(filtered) - BG(3);

figure;
subplot(2,1,1);
line(cfp,fret,'Linestyle','none','Marker','.','MarkerSize',5,'color','k');
x1 = linspace(0,max(cfp),1000);
line(x1,polyval(RD1,x1),'color','r','linewidth',2);
line(cfp,fret-polyval(RD1,cfp),'Linestyle','none','Marker','.','MarkerSize',5,'color','b');
line(x1,zeros(size(x1)),'linestyle','--','color','k');
xlabel('cfp');ylabel('fret');title(listboxstr(listboxval));
subplot(2,1,2);
x2 = linspace(0,max(yfp),1000);
line(yfp,fret,'Linestyle','none','Marker','.','MarkerSize',5,'color','k');
line(x2,polyval(RA,x2),'color','r','linewidth',2);
line(x2,zeros(size(x2)),'linestyle','--','color','k');
xlabel('yfp');ylabel('fret');title(listboxstr(listboxval));
% --------------------------------------------------------------------
function setMAMD_Callback(hObject, eventdata, handles)
linkaxes([handles.axes6 handles.axes7],'off');
linkaxes([handles.axes8 handles.axes9],'off');
DATA = load(handles.analysisfile,'-mat');
listboxstr = get(handles.construct_listbox,'string');
listboxval = get(handles.construct_listbox,'value');
if length(listboxval)>1
    errordlg('Please select only 1 construct at a time');
    return;
end
conname = listboxstr{listboxval};
conindex = find(strcmp({DATA.constructs.name},conname));
handles.conindex = conindex;
if ~isfield(DATA,'constants')
    errordlg('Set BG, RA, RD first');
    return;
else
    if ~isfield(DATA.constants,'BGi') | ~isfield(DATA.constants,'RAi') | ~isfield(DATA.constants,'RDi')
        errordlg('Set BG, RA, RD first');
        return;
    end
end
```

```
if ~strcmp(get(gcbo,'tag'),'setMAMD')
    listboxstr = get(handles.listbox,'String');
    listboxval = get(handles.listbox,'Value');
    for k=1:length(listboxval)
        sel = listboxstr{listboxval(k)};
        fileindex(k) = find(strcmp({DATA.constructs(conindex).datafile.name},sel));
        CVnames(k,:) = DATA.constructs(conindex).datafile(fileindex(k)).CVnames;
    end
else
    dates = {DATA.constants.date};
    [chosendate,v] = listdlg('PromptString','Associate with following date:',...
                'SelectionMode','single',...
                'ListString',dates);

    if ~v
        return;
    end
    chosendate = dates{chosendate};

    if isempty(DATA.constructs(conindex).datafile(1).name)
        datafilelength = 0;
    else
        datafilelength = length(DATA.constructs(conindex).datafile);
    end

    DATA.constructs(conindex).datafile(datafilelength + 1).name = chosendate;

    listboxstr = [];
    for k=1:length(DATA.constructs(conindex).datafile)
        listboxstr{k} = DATA.constructs(conindex).datafile(k).name;
    end
    set(handles.listbox,'string',listboxstr,'Max',length(listboxstr),'Value',length(listboxstr));

    fileindex = length(listboxstr);

    % add files to this entry
    % list all constructs
    names = {};
    for k=1:length(DATA.constructs)
        names = [names, {DATA.constructs(k).datafile.name}];
    end
    filesind = listdlg('PromptString','Select CV constructs:', ...
        'SelectionMode','multiple','ListString',names);

    CVnames = names(filesind);
    DATA.constructs(conindex).datafile(fileindex).CVnames = CVnames;
    DATA.constructs(conindex).plottype = 'MAMD';
end

for z = 1:size(CVnames,1)
    cindex = zeros(size(CVnames(z,:)));
    findex = zeros(size(CVnames(z,:)));
    map = jet;


    for m=1:size(CVnames,2)

        for k=1:length(DATA.constructs)
             temp = find(strcmp(CVnames(z,m),{DATA.constructs(k).datafile.name}));
             if ~isempty(temp)
                 findex(m) = temp;
                 cindex(m) = k;
             end
        end

        filename = DATA.constructs(cindex(m)).datafile(findex(m)).filename;
        filenameindex = strfind(filename,'FLOW');   % hack to include older files from SRL's computer
        if ~isempty(filenameindex)
            filename = [handles.dir, filename(filenameindex+4:end)];
        else
            filename = [handles.dir, filename];
        end
        [fcsdat, fcshdr] = fca_readfcs(filename);

        if isempty(fcshdr)
            errordlg('Header file cannot be read');
            return;
        end
        %
        % define the parameter names
        %
        if strfind(fcshdr.cytometry,'Attune')
            handles.saturation = 1e7;
            handles.CFPname = 'VL1-A';
            handles.YFPname = 'BL1-A';
            handles.FRETname = 'VL2-A';
            handles.Cherryname = 'BL3-A';
            handles.singlet1x = 'FSCA';
            handles.singlet1y = 'FSCH';
            handles.singlet2x = 'SSCA';
            handles.singlet2y = 'SSCH';
```

```
else
    handles.saturation = 262000;
    handles.CFPname = 'pacific Blue-A';
    handles.YFPname = 'FITC-A';
    handles.FRETname = 'AmCyan-A';
    handles.singlet1x = 'FSCH';
    handles.singlet1y = 'FSCW';
    handles.singlet2x = 'SSCH';
    handles.singlet2y = 'SSCW';
end

set(handles.filename_editbox,'String',fcshdr.filename);
parnames_long = cell(1,fcshdr.NumOfPar);
parnames = cell(1,fcshdr.NumOfPar);
for i=1:fcshdr.NumOfPar
    parnames(i) = {fcshdr.par(i).name};
end
handles.parnames = parnames;
set(handles.numofevent_edit,'String',fcshdr.TotalEvents);
set(handles.numinroi_edit,'String',fcshdr.TotalEvents);
handles.TotalEvents = fcshdr.TotalEvents;
if isempty(find(strcmpi(handles.CFPname,handles.parnames)))
    cfpvalue = listdlg('PromptString','Select CFP channel:',...
                'SelectionMode','single',...
                'ListString',handles.parnames);
    handles.CFPname = handles.parnames{cfpvalue};
    yfpvalue = listdlg('PromptString','Select YFP channel:',...
                'SelectionMode','single',...
                'ListString',handles.parnames);
    handles.YFPname = handles.parnames{yfpvalue};
    fretvalue = listdlg('PromptString','Select FRET channel:',...
                'SelectionMode','single',...
                'ListString',handles.parnames);
    handles.FRETname = handles.parnames{fretvalue};
else
    cfpvalue = find(strcmpi(handles.CFPname,handles.parnames));
    yfpvalue = find(strcmpi(handles.YFPname,handles.parnames));
    fretvalue = find(strcmpi(handles.FRETname,handles.parnames));
    cherryvalue = find(strcmpi(handles.Cherryname,handles.parnames));
end
set(handles.cfpchan_popup,'string',handles.CFPname);
set(handles.yfpchan_popup,'string',handles.YFPname);
set(handles.fretchan_popup,'string',handles.FRETname);

set(handles.cfpPMT,'string',fcshdr.par(cfpvalue).PMTV);
set(handles.yfpPMT,'string',fcshdr.par(yfpvalue).PMTV);
set(handles.fretPMT,'string',fcshdr.par(fretvalue).PMTV);

handles.FSCPMTV = fcshdr.par(find(strcmpi('FSC-A',handles.parnames))).PMTV;
handles.SSCPMTV = fcshdr.par(find(strcmpi('SSC-A',handles.parnames))).PMTV;
handles.cfpPMTV = fcshdr.par(cfpvalue).PMTV;
handles.yfpPMTV = fcshdr.par(yfpvalue).PMTV;
handles.fretPMTV = fcshdr.par(fretvalue).PMTV;
handles.cherryPMTV = fcshdr.par(cherryvalue).PMTV;

handles.FSCA = fcsdat(:,find(strcmpi('FSC-A',handles.parnames)));
handles.SSCA = fcsdat(:,find(strcmpi('SSC-A',handles.parnames)));
handles.FSCH = fcsdat(:,find(strcmpi('FSC-H',handles.parnames)));
handles.FSCW = fcsdat(:,find(strcmpi('FSC-W',handles.parnames)));
handles.SSCH = fcsdat(:,find(strcmpi('SSC-H',handles.parnames)));
handles.SSCW = fcsdat(:,find(strcmpi('SSC-W',handles.parnames)));
handles.cfp = fcsdat(:,cfpvalue);
handles.yfp = fcsdat(:,yfpvalue);
handles.fret = fcsdat(:,fretvalue);
handles.cherry = fcsdat(:,cherryvalue);
handles.fcsdate = fcshdr.date;
% scale fluorescence values to the same PMTV for Attune
if strfind(fcshdr.cytometry,'Attune')
    load([handles.dir '\PMTcalib.mat']);
    % scale FSC
    defaultPMTV = 1750;
    currentPMTV = str2num(handles.FSCPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(FSCHc),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.FSCA = handles.FSCA*scalefactor;
    handles.FSCH = handles.FSCH*scalefactor;

    % scale SSC
    defaultPMTV = 2250;
    currentPMTV = str2num(handles.SSCPMTV);
    xi = [currentPMTV defaultPMTV];
    yi = exp(interp1(log(PMTVc),log(SSCHc),log(xi)));
    scalefactor = yi(2)/yi(1);
    handles.SSCA = handles.SSCA*scalefactor;
    handles.SSCH = handles.SSCH*scalefactor;

    % scale BL-1
    defaultPMTV = 1000;
    currentPMTV = str2num(handles.yfpPMTV);
    xi = [currentPMTV defaultPMTV];
```

```
            yi = exp(interp1(log(PMTVc),log(BL1Ac),log(xi)));
            scalefactor = yi(2)/yi(1);
            handles.yfp = handles.yfp*scalefactor;

            % scale VL-1
            defaultPMTV = 1000;
            currentPMTV = str2num(handles.cfpPMTV);
            xi = [currentPMTV defaultPMTV];
            yi = exp(interp1(log(PMTVc),log(VL1Ac),log(xi)));
            scalefactor = yi(2)/yi(1);
            handles.cfp = handles.cfp*scalefactor;

            % scale VL-2
            defaultPMTV = 1000;
            currentPMTV = str2num(handles.fretPMTV);
            xi = [currentPMTV defaultPMTV];
            yi = exp(interp1(log(PMTVc),log(VL2Ac),log(xi)));
            scalefactor = yi(2)/yi(1);
            handles.fret = handles.fret*scalefactor;

            % scale BL-3
            defaultPMTV = 1000;
            currentPMTV = str2num(handles.cherryPMTV);
            xi = [currentPMTV defaultPMTV];
            yi = exp(interp1(log(PMTVc),log(BL3Ac),log(xi)));
            scalefactor = yi(2)/yi(1);
            handles.cherry = handles.cherry*scalefactor;

        else
            warndlg('PMTVs are not scaled. Do not have PMT calibration data for this cytometer');
        end



        % filter based on previous singlet discrimination, and current filter1
%         filter1 = get(handles.filter1_edit,'string');
        if ~isfield(DATA.constructs(conindex).datafile(fileindex(z)),'filter1')
            filter1 = '';
        else
            filter1 = DATA.constructs(conindex).datafile(fileindex(z)).filter1;
        end
        set(handles.filter1_edit,'String',filter1);


        handles.roi1pos = DATA.constructs(cindex(m)).datafile(findex(m)).roi1;
        handles.roi1index = inpolygon(handles.FSCA,handles.SSCA,handles.roi1pos(:,1),handles.roi1pos(:,2));

        handles.roi2pos = DATA.constructs(cindex(m)).datafile(findex(m)).roi2;
        handles.roi2index =
inpolygon(handles.(handles.singlet1x),handles.(handles.singlet1y),handles.roi2pos(:,1),handles.roi2pos(:,2));

        handles.roi3pos = DATA.constructs(cindex(m)).datafile(findex(m)).roi3;
        handles.roi3index =
inpolygon(handles.(handles.singlet2x),handles.(handles.singlet2y),handles.roi3pos(:,1),handles.roi3pos(:,2));


        % written filters
        cfp = handles.cfp;
        yfp = handles.yfp;
        fret = handles.fret;
        if ~isempty(filter1)
            handles.filter1index = eval(filter1);
        else
            handles.filter1index = ones(handles.TotalEvents,1);
        end

        % do not display saturated and non-zero events
        nonsatnonzero = (handles.FSCA<handles.saturation & handles.SSCA<handles.saturation & handles.FSCA>0 &
handles.SSCA>0);
        nonsatnonzero = nonsatnonzero & cfp<handles.saturation & yfp<handles.saturation & fret<handles.saturation & ...
                cfp>0 & yfp>0 & fret>0;
        singlets = nonsatnonzero & handles.roi1index & handles.roi2index & handles.roi3index;
        filtered1 = nonsatnonzero & handles.roi1index & handles.roi2index & handles.roi3index & handles.filter1index;

    %     set(handles.numinroi_edit,'string',num2str(sum(filtered1)));
    %     set(handles.numinroi2_edit,'string',num2str(sum(filtered2)));

        if length(CVnames)>1
            color = map(round(1+(m-1)/(length(CVnames)-1)*(length(map)-1)),:);
        else
            color = 'k';
        end


        cfp = cfp(find(filtered1));
        yfp = yfp(find(filtered1));
        fret = fret(find(filtered1));

        % get constants for this date
        date = handles.fcsdate;
        constantsindex = find(strcmp(date,{DATA.constants.date}));
```

```
        if isempty(DATA.constants(constantsindex).BGi) | isempty(DATA.constants(constantsindex).RAi) ...
                | isempty(DATA.constants(constantsindex).RDi)
            errordlg('Please set constants first');
            return;
        end
        ccindex = DATA.constants(constantsindex).BGi(1);
        cfindex = DATA.constants(constantsindex).BGi(2);

        BG = DATA.constructs(ccindex).datafile(cfindex).BG;
        cfp = cfp - BG(1);
        yfp = yfp - BG(2);
        fret = fret - BG(3);

        ccindex = DATA.constants(constantsindex).RAi(1);
        cfindex = DATA.constants(constantsindex).RAi(2);

        RAl = DATA.constructs(ccindex).datafile(cfindex).RAl;
%         RAh = DATA.constructs(ccindex).datafile(cfindex).RAh;
        RAthres = DATA.constructs(ccindex).datafile(cfindex).RAthres;
        set(handles.RA_edit,'string',num2str(RAl(1)));

        ccindex = DATA.constants(constantsindex).RDi(1);
        cfindex = DATA.constants(constantsindex).RDi(2);

        RD1l = DATA.constructs(ccindex).datafile(cfindex).RD1l;
%         RD1h = DATA.constructs(ccindex).datafile(cfindex).RD1h;
        RD1thres = DATA.constructs(ccindex).datafile(cfindex).RD1thres;
        set(handles.RD1_edit,'string',num2str(RD1l(1)));

        RD2l = DATA.constructs(ccindex).datafile(cfindex).RD2l;
%         RD2h = DATA.constructs(ccindex).datafile(cfindex).RD2h;
        set(handles.RD2_edit,'string',num2str(RD2l(1)));

        underyfpthres = (yfp<=RAthres);
        overyfpthres = (yfp>RAthres);

        undercfpthres = (cfp<=RD1thres);
        overcfpthres = (cfp>RD1thres);

        ind = find(underyfpthres & undercfpthres);
        RA = RAl; RD1 = RD1l; RD2 = RD2l;

        YFPd = polyval(RA,(yfp(ind)-polyval(RD2,cfp(ind))));
        CFPd = polyval(RD1,cfp(ind));
        YFPf = fret(ind) - polyval(RD1,cfp(ind)) - YFPd;

        x1 = YFPd./CFPd;
        y1 = YFPf./CFPd;

        x = x1; y = y1;
        ind = find(x>0 & x<0.1);
        xf = x(ind); yf = y(ind);
        sl = xf\yf;
        axes(handles.axes6);
        line(x,y,'linestyle','none','marker','.','markersize',3,'color',color,'tag','data');
        xs = [0 0.06];
        line(xs,xs*sl,'linestyle','-','color',color,'tag','data');
        xx(m) = median(x); yy(m) = median(y);
%         xx(m) = trimmean(x,0.1); yy(m) = trimmean(y,0.1);

        % also plot estimates of NA/ND if GAGD and FAFD available

        GAGDn = DATA.constructs(conindex).datafile(fileindex(z)).GAGD;
        FAFDn = DATA.constructs(conindex).datafile(fileindex(z)).FAFD;
        if ~isempty(GAGDn) & ~isempty(FAFDn)
            % estimate NA/ND, Eeff33/EeffDD.
            Eeff33 = GAGDn*YFPf./YFPd;
            EeffDD = YFPf./(YFPf + CFPd*FAFDn);
            NAND = YFPd.*(1-Eeff33)./(CFPd*GAGDn*FAFDn);

            xmin = 5000;%150;
            xmax = 30000;%400;

            axes(handles.axes7);
            ylabel('Eeff33'); xlabel('YFPest');
            line(YFPd/(GAGDn*FAFDn),Eeff33,'linestyle','none','marker','.','markersize',5,'color',color,'tag','data');
            set(gca,'XLim',[xmin xmax],'YLim',[0 0.7]);

            axes(handles.axes9);
            ylabel('EeffDD'); xlabel('YFPest');
            line(YFPd/(GAGDn*FAFDn),EeffDD,'linestyle','none','marker','.','markersize',5,'color',color,'tag','data');
            set(gca,'XLim',[xmin xmax],'YLim',[0 0.7]);

            axes(handles.axes8);
            ylabel('NA/ND'); xlabel('YFPd');
            line(YFPd,NAND,'linestyle','none','marker','.','markersize',5,'color',color,'tag','data');
            set(gca,'XLim',[xmin xmax],'YLim',[0 2]);

            ind = find(YFPd>xmin & YFPd<xmax);
            meanEeff33 = trimmean(Eeff33(ind),1);
```

```
            stdEeff33 = std(Eeff33(ind));
            meanEeffDD = trimmean(EeffDD(ind),1);
            stdEeffDD = std(EeffDD(ind));
            meanNAND = trimmean(NAND(ind),1);
            stdNAND = std(NAND(ind));

            axes(handles.axes4);
            hold on;
            errorbar(2*m-
1,meanEeff33,stdEeff33,'linestyle','none','marker','.','markersize',10,'color',color,'tag','data');

errorbar(2*m,meanEeffDD,stdEeffDD,'linestyle','none','marker','.','markersize',10,'color',color,'tag','data');
            ylabel('Eeff33 / EeffDD');
            ylim([0 0.7]); xlim([0 2*m+1]);
            set(gca,'XScale','linear','YScale','linear');
            axes(handles.axes5);
            hold on;
            errorbar(m,meanNAND,stdNAND,'linestyle','none','marker','.','markersize',10,'color',color,'tag','data');
            ylabel('NA/ND');
            ylim([0 2]); xlim([0 m+1]);
            set(gca,'XScale','linear','YScale','linear');
        end
    end

%     line(xx,yy,'linestyle','none','marker','.','markersize',20,'tag','data','color','r');
%     xlim([min(xx)*0.5 max(xx)*1.3]);
%     ylim([min(yy)*0.1 max(yy)*1.3]);

    [p,S] = polyfit(xx,yy,1);
    p(1) = 56.06952621;
%     p(2) = -1.77555;
    p(2) = -1.65;
    xxx = [0,0.2];
    yyy = xxx*p(1) + p(2);
    axes(handles.axes6);

    line(xxx,yyy,'color','k','linewidth',2,'tag','data');
    set(handles.axes6,'XLim',[0 0.06], 'YLim', [-0.1 1.8]);
    GaGd = 1/p(1);
    FaFd = -p(2);
    text(0.1, 0.8, {'Default:'; ['GA/GD = ' num2str(GaGd)]; ['FA/FD = '
num2str(FaFd)]},'tag','data','Units','Normalized');
    GAGDn = DATA.constructs(conindex).datafile(fileindex(z)).GAGD;
    FAFDn = DATA.constructs(conindex).datafile(fileindex(z)).FAFD;
    if ~isempty(GAGDn) & ~isempty(FAFDn)
        set(handles.GAGD_edit,'string',num2str(GAGDn));
        set(handles.FAFD_edit,'string',num2str(FAFDn));
        line(xxx, xxx/GAGDn - FAFDn,  'color','r','linewidth',2,'tag','data');
    else
        DATA.constructs(conindex).datafile(fileindex(z)).GAGD = GaGd;
        DATA.constructs(conindex).datafile(fileindex(z)).FAFD = FaFd;
        set(handles.GAGD_edit,'string',num2str(GaGd));
        set(handles.FAFD_edit,'string',num2str(FaFd));
    end
end

% clear DATA;
% DATA.GA = GD*GaGd;
% DATA.FA = FD*FaFd;

save(handles.analysisfile,'-struct','DATA','-append');
handles.confilelength = length(get(handles.listbox,'string'));
guidata(gcf,handles);



% --------------------------------------------------------------------
function changeGD_Callback(hObject, eventdata, handles)
DATA = load(handles.analysisfile,'GA','GD');
GaGd = DATA.GA/DATA.GD;
prompt = {'Enter new GD'};
dlg_title = 'Change GD';
num_lines = 1;
oldGD = DATA.GD;
def = {num2str(oldGD)};
answer = inputdlg(prompt,dlg_title,num_lines,def);
DATA.GD = str2num(answer{1});
DATA.GA = DATA.GD*GaGd;
ratio = DATA.GD/oldGD;
save(handles.analysisfile,'-struct','DATA','-append');

listboxstr = get(handles.listbox,'string');
listboxval = listdlg('PromptString','Select constructs to update:', ...
    'SelectionMode','multiple','ListString',listboxstr);
h = waitbar(0,'Please wait...');

for k=1:length(listboxval)
    waitbar(k/length(listboxval),h);
    clear DATA;
    set(handles.listbox,'value',listboxval(k));
    structname = ['STR', num2str(listboxval(k),'%04.0f')];
```

```
    DATA = load(handles.analysisfile,structname);
    DATA.(structname).KD = DATA.(structname).KD/ratio;
    DATA.(structname).KD2 = DATA.(structname).KD2/ratio;
    save(handles.analysisfile,'-struct','DATA','-append');
end
close(h);
% --------------------------------------------------------------------
function viewconstants_Callback(hObject, eventdata, handles)
load(handles.analysisfile,'RA','RD1','RD2','GA','GD','FA','FD');
msgbox({['RA = ' num2str(RA)]; ...
    ['RD1 = ' num2str(RD1)]; ...
    ['RD2 = ' num2str(RD2)]; ...
    ['GA = ' num2str(GA)]; ...
    ['GD = ' num2str(GD)]; ...
    ['FA = ' num2str(FA)]; ...
    ['FD = ' num2str(FD)]});

% --------------------------------------------------------------------
function exportfiltered_Callback(hObject, eventdata, handles)


% --------------------------------------------------------------------
function keeprois_Callback(hObject, eventdata, handles)
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);


function UpdateMAMD
handles = guidata(gcf);
clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');

DATA = load(handles.analysisfile,'-mat');
if isfield(handles,'fcsdate') & ~strcmp(clistboxstr{clistboxval},'MAMD')
    date = handles.fcsdate;
    constantsindex = find(strcmp(date,{DATA.constants.date}));

    ccindex = DATA.constants(constantsindex).MAMDi(1);
    cfindex = DATA.constants(constantsindex).MAMDi(2);

else

        confilelength = handles.confilelength;
        listboxstr = get(handles.listbox,'String');
        listboxval = get(handles.listbox,'Value');
        sel = listboxstr{listboxval};
        if length(listboxval)>1
            errordlg('Please select only one file at a time');
            return;
        end
        cval = find(listboxval<=cumsum(confilelength),1,'first');
        conname = clistboxstr{cval};
        ccindex = clistboxval(find(strcmp({DATA.constructs.name},conname)));
        cfindex = find(strcmp({DATA.constructs(ccindex).datafile.name},sel));
    if ~strcmp(clistboxstr{clistboxval},'MAMD')
%         filename = DATA.constructs(ccindex).datafile(cfindex).filename;
%         [fcsdat, fcshdr] = fca_readfcs(filename);
%
        errordlg('Can''t find associated MAMD file to update');
        return;
    end
end

GAGD = str2num(get(handles.GAGD_edit,'string'));
FAFD = str2num(get(handles.FAFD_edit,'string'));

DATA.constructs(ccindex).datafile(cfindex).GAGD = GAGD;
DATA.constructs(ccindex).datafile(cfindex).FAFD = FAFD;
save(handles.analysisfile,'-struct','DATA','-append');

listbox_Callback(handles.listbox,[],handles);

function UpdateRARD
handles = guidata(gcf);
DATA = load(handles.analysisfile,'-mat');

clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');
confilelength = handles.confilelength;
listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');
sel = listboxstr{listboxval};
if length(listboxval)>1
    errordlg('Please select only one file at a time');
    return;
end
cval = find(listboxval<=cumsum(confilelength),1,'first');
```

```
conname = clistboxstr{cval};
if strcmpi(clistboxstr{clistboxval},'Cer')
    ccindexRD = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    cfindexRD = find(strcmp({DATA.constructs(ccindexRD).datafile.name},sel));
elseif strcmpi(clistboxstr{clistboxval},'Ven')
    ccindexRA = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    cfindexRA = find(strcmp({DATA.constructs(ccindexRA).datafile.name},sel));
elseif strcmpi(clistboxstr{clistboxval},'Cherry')
    ccindexRC = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    cfindexRC = find(strcmp({DATA.constructs(ccindexRC).datafile.name},sel));
else
    if isfield(handles,'fcsdate')
        date = handles.fcsdate;
        constantsindex = find(strcmp(date,{DATA.constants.date}));
        if ~isempty(DATA.constants(constantsindex).RAi)
            ccindexRA = DATA.constants(constantsindex).RAi(1);
            cfindexRA = DATA.constants(constantsindex).RAi(2);
        end
        if ~isempty(DATA.constants(constantsindex).RDi)
            ccindexRD = DATA.constants(constantsindex).RDi(1);
            cfindexRD = DATA.constants(constantsindex).RDi(2);
        end
        if ~isempty(DATA.constants(constantsindex).RCi)
            ccindexRC = DATA.constants(constantsindex).RCi(1);
            cfindexRC = DATA.constants(constantsindex).RCi(2);
        end
    else
        errordlg('Can''t find associated file to update');
        return;
    end
end


RA = str2num(get(handles.RA_edit,'string'));
RA2 = str2num(get(handles.RA2_edit,'string'));
RD1 = str2num(get(handles.RD1_edit,'string'));
RD2 = str2num(get(handles.RD2_edit,'string'));
RD3 = str2num(get(handles.RD3_edit,'string'));
RC1 = str2num(get(handles.RC1_edit,'string'));
RC2 = str2num(get(handles.RC2_edit,'string'));
RC3 = str2num(get(handles.RC3_edit,'string'));

if exist('ccindexRA','var')
    DATA.constructs(ccindexRA).datafile(cfindexRA).RAl(1) = RA;
    DATA.constructs(ccindexRA).datafile(cfindexRA).RA2(1) = RA2;
    save(handles.analysisfile,'-struct','DATA','-append');
end

if exist('ccindexRD','var')
    DATA.constructs(ccindexRD).datafile(cfindexRD).RD1l(1) = RD1;
    DATA.constructs(ccindexRD).datafile(cfindexRD).RD2l(1) = RD2;
    DATA.constructs(ccindexRD).datafile(cfindexRD).RD3(1) = RD3;
    save(handles.analysisfile,'-struct','DATA','-append');
end

if exist('ccindexRC','var')
    DATA.constructs(ccindexRC).datafile(cfindexRC).RC1(1) = RC1;
    DATA.constructs(ccindexRC).datafile(cfindexRC).RC2(1) = RC2;
    DATA.constructs(ccindexRC).datafile(cfindexRC).RC3(1) = RC3;
    save(handles.analysisfile,'-struct','DATA','-append');
end
listbox_Callback(handles.listbox,[],handles);

function UpdateFlowParams
handles = guidata(gcf);
DATA = load(handles.analysisfile,'-mat');
clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');
confilelength = handles.confilelength;

listboxstr = get(handles.listbox,'String');
listboxval = get(handles.listbox,'Value');
cla(handles.axes6); cla(handles.axes7);
cla(handles.axes8); cla(handles.axes9);

for z=1:length(listboxval)
    sel = listboxstr{listboxval(z)};
    cval = find(listboxval(z)<=cumsum(confilelength),1,'first');
    conname = clistboxstr{cval};
    conindex = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    fileindex = find(strcmp({DATA.constructs(conindex).datafile.name},sel));

    KD = str2num(get(handles.KD,'string'));
    KD2 = str2num(get(handles.KD2,'string'));
    spurFR = str2num(get(handles.spurFR,'string'));
    spurEFRET = str2num(get(handles.spurEFRET,'string'));
    EFRmax = str2num(get(handles.EFRmax,'string'));
    EFRETmax = str2num(get(handles.EFRETmax,'string'));

    DATA.constructs(conindex).datafile(fileindex).KD = KD;
    DATA.constructs(conindex).datafile(fileindex).KD2 = KD2;
```

```
    DATA.constructs(conindex).datafile(fileindex).spurFR = spurFR;
    DATA.constructs(conindex).datafile(fileindex).spurEFRET = spurEFRET;
    DATA.constructs(conindex).datafile(fileindex).EFRmax = EFRmax;
    DATA.constructs(conindex).datafile(fileindex).EFRETmax = EFRETmax;

    save(handles.analysisfile,'-struct','DATA','-append');
    UpdateFlowFretC(2);
end



% --------------------------------------------------------------------
function showfiltered_Callback(hObject, eventdata, handles)
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);
% --------------------------------------------------------------------
function showExp_Callback(hObject, eventdata, handles)
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);
% --------------------------------------------------------------------
function plotvsred_Callback(hObject, eventdata, handles)
% hObject    handle to plotvsred (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);

% --------------------------------------------------------------------
function R2C2adjust_Callback(hObject, eventdata, handles)
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);

% --------------------------------------------------------------------
function plotvstime_Callback(hObject, eventdata, handles)
% hObject    handle to plotvstime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in bin.
if strcmpi(get(hObject,'Checked'),'on')
    set(hObject,'Checked','off');
else
    set(hObject,'Checked','on');
end
guidata(hObject,handles);


function bin_Callback(hObject, eventdata, handles)
xbin = get(handles.xbin_edit,'string');
if isempty(xbin)
    errordlg('Please input x bin coordinates');
    return;
end
xbin = str2num(xbin);
datah = findobj(handles.axes6,'type','line','marker','.');
xdataall = get(datah,'xdata');
ydataall = get(datah,'ydata');

for z=1:length(datah)
    if iscell(xdataall)
        xdata = xdataall{z};
        ydata = ydataall{z};
    else
        xdata = xdataall;
        ydata = ydataall;
    end
    clear bmeanx bmeany bstdx bstdy bstex bstey numcells

    for k=1:length(xbin)-1
        indices = find(xdata>xbin(k) & xdata<xbin(k+1));
        perc = 0.05;
        newindices = ceil(length(indices)*perc/2) : floor(length(indices)*(1-perc/2));
        if length(newindices)>2
            goodmean(k) = trimmean(ydata(indices),5);
            [sorty,IX] = sort(ydata(indices));
            sortx = xdata(indices);
```

```
            sortx = sortx(IX);

            reducedy = sorty(newindices);
            reducedx = sortx(newindices);
            bmeany(k) = mean(reducedy);
            bmeanx(k) = mean(reducedx);
            numcells(k) = length(newindices);
            bstex(k) = std(reducedx); %/sqrt(numcells(k));
            bstey(k) = std(reducedy); %/sqrt(numcells(k));
            addtonext = 0;
        else
            numcells(k) = NaN;
            bmeanx(k) = NaN;
            bmeany(k) = NaN;
            bstex(k) = NaN;
            bstey(k) = NaN;
        end
    end
    set(datah(z),'xdata',bmeanx,'ydata',bmeany,'MarkerSize',20);
    Copy2Clipboard([bmeanx', bmeany']);
    axes(handles.axes6);
    % draw error bars (std)
    for k=1:length(xbin)-1
        if (~isnan(bstex(k)) & ~isnan(bstey(k)))
            line([bmeanx(k) bmeanx(k)], [bmeany(k)-bstey(k) bmeany(k)+bstey(k)],'color','k','tag','data'); % yerror
            line([bmeanx(k)-bstex(k) bmeanx(k)+bstex(k)], [bmeany(k) bmeany(k)],'color','k','tag','data'); % xerror
        end
    end
%     xlim([min(bmeanx)*0.9 max(bmeanx)*1.1]);
%     ylim([-0.1*max(bmeany) 1.1*max(bmeany)]);
    % errorbarsxy(bmeanx, bmeany, bstex, bstey,'k');

end


% --------------------------------------------------------------------
function compiledata_Callback(hObject, eventdata, handles)

DATA = load(handles.analysisfile,'-mat');
clistboxstr = get(handles.construct_listbox,'string');
clistboxval = get(handles.construct_listbox,'value');
confilelength = handles.confilelength;

listboxstr = get(handles.listbox,'string');
listboxval = listdlg('PromptString','Select constructs:', ...
    'SelectionMode','multiple','ListString',listboxstr);

for k=1:length(listboxval)
    sel = listboxstr{listboxval(k)};
    cval = find(listboxval(k)<=cumsum(confilelength),1,'first');
    conname = clistboxstr{cval};
    conindex = clistboxval(find(strcmp({DATA.constructs.name},conname)));
    fileindex = find(strcmp({DATA.constructs(conindex).datafile.name},sel));

    construct{k,1} = DATA.constructs(conindex).datafile(fileindex).name;
    KD(k,1) = DATA.constructs(conindex).datafile(fileindex).KD;
    Emax(k,1) = DATA.constructs(conindex).datafile(fileindex).EFRmax;
    KDstd(k,1) = DATA.constructs(conindex).datafile(fileindex).KDstd;
    Emaxstd(k,1) = DATA.constructs(conindex).datafile(fileindex).EFRmaxstd;
    KD2(k,1) = DATA.constructs(conindex).datafile(fileindex).KD2;
    Emax2(k,1) = DATA.constructs(conindex).datafile(fileindex).EFRETmax;
    KD2std(k,1) = DATA.constructs(conindex).datafile(fileindex).KD2std;
    Emax2std(k,1) = DATA.constructs(conindex).datafile(fileindex).EFRETmaxstd;
    FRNumCells(k,1) = DATA.constructs(conindex).datafile(fileindex).FRnumcells;
    EFRETNumCells(k,1) = DATA.constructs(conindex).datafile(fileindex).EFRETnumcells;
end

header = {'Construct', 'KD', 'Emax', 'KDstd', 'Emaxstd','FRNumCells',...
                      'KD2', 'Emax2', 'KD2std', 'Emax2std', 'EFRETNumCells'};
Copy2Clipboard([header ; [construct, num2cell(KD), num2cell(Emax), num2cell(KDstd), num2cell(Emaxstd),
num2cell(FRNumCells),...
    num2cell(KD2), num2cell(Emax2), num2cell(KD2std), num2cell(Emax2std), num2cell(EFRETNumCells)]])
msgbox('Copied to Clipboard');
guidata(hObject,handles);




function bin2_Callback(hObject, eventdata, handles)
xbin = get(handles.xbin2_edit,'string');
if isempty(xbin)KD_
    errordlg('Please input x bin coordinates');
    return;
end
xbin = str2num(xbin);
datah = findobj(handles.axes8,'type','line','marker','.');
xdataall = get(datah,'xdata');
ydataall = get(datah,'ydata');

clear bmeanx bmeany bstdx bstdy bstex bstey numcells
```

```
for z=1:length(datah)
    if iscell(xdataall)
        xdata = xdataall{z};
        ydata = ydataall{z};
    else
        xdata = xdataall;
        ydata = ydataall;
    end
    clear bmeanx bmeany bstdx bstdy bstex bstey numcells

    for k=1:length(xbin)-1
        indices = find(xdata>xbin(k) & xdata<xbin(k+1));
        perc = 0.05;
        newindices = ceil(length(indices)*perc/2) : floor(length(indices)*(1-perc/2));
        if length(newindices)>2
            goodmean(k) = trimmean(ydata(indices),5);
            [sorty,IX] = sort(ydata(indices));
            sortx = xdata(indices);
            sortx = sortx(IX);

            reducedy = sorty(newindices);
            reducedx = sortx(newindices);
            bmeany(k) = mean(reducedy);
            bmeanx(k) = mean(reducedx);
            numcells(k) = length(newindices);
            bstex(k) = std(reducedx); %/sqrt(numcells(k));
            bstey(k) = std(reducedy); %/sqrt(numcells(k));
            addtonext = 0;
        else
            numcells(k) = NaN;
            bmeanx(k) = NaN;
            bmeany(k) = NaN;
            bstex(k) = NaN;
            bstey(k) = NaN;
        end
    end
    set(datah(z),'xdata',bmeanx,'ydata',bmeany,'MarkerSize',20);

    axes(handles.axes8);
    % draw error bars (actually std)
    for k=1:length(xbin)-1
        if (~isnan(bstex(k)) & ~isnan(bstey(k)))
            line([bmeanx(k) bmeanx(k)], [bmeany(k)-bstey(k) bmeany(k)+bstey(k)],'color','k','tag','data'); % yerror
            line([bmeanx(k)-bstex(k) bmeanx(k)+bstex(k)], [bmeany(k) bmeany(k)],'color','k','tag','data'); % xerror
        end
    end
%     xlim([min(bmeanx)*0.9 max(bmeanx)*1.1]);
%     ylim([-0.1*max(bmeany) 1.1*max(bmeany)]);
    % errorbarsxy(bmeanx, bmeany, bstex, bstey,'k');

end

function xbin2_edit_Callback(hObject, eventdata, handles)



function filter1_edit_Callback(hObject, eventdata, handles)
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ', ''Tag'', ''data''));'])
    eval(['delete(findobj(handles.axes' num2str(k) ', ''type'',''text''));'])
end
UpdateFlowFretC(2);

function filter2_edit_Callback(hObject, eventdata, handles)
for k=4:9
    eval(['delete(findobj(handles.axes' num2str(k) ', ''Tag'', ''data''));'])
    eval(['delete(findobj(handles.axes' num2str(k) ', ''type'',''text''));'])
end
UpdateFlowFretC(2);

% --------------------------------------------------------------------
function fitEFRET1_Callback(hObject, eventdata, handles)
handles.tryfitEFRET1 = 1;
guidata(hObject,handles);
cla(handles.axes6);cla(handles.axes7);
cla(handles.axes8);cla(handles.axes9);
UpdateFlowFretC;
handles.tryfitEFRET1 = 0;
guidata(hObject,handles);

% --------------------------------------------------------------------
function fitEFRET2_Callback(hObject, eventdata, handles)
handles.tryfitEFRET2 = 1;
guidata(hObject,handles);
cla(handles.axes6);cla(handles.axes7);
cla(handles.axes8);cla(handles.axes9);
UpdateFlowFretC;
handles.tryfitEFRET2 = 0;
guidata(hObject,handles);

% --------------------------------------------------------------------
```

```
function fitFR1_Callback(hObject, eventdata, handles)
handles.tryfitFR1 = 1;
guidata(hObject,handles);
cla(handles.axes6);cla(handles.axes7);
cla(handles.axes8);cla(handles.axes9);
UpdateFlowFretC;
handles.tryfitFR1 = 0;
guidata(hObject,handles);


% --------------------------------------------------------------------
function fitFR2_Callback(hObject, eventdata, handles)
handles.tryfitFR2 = 1;
guidata(hObject,handles);
cla(handles.axes6);cla(handles.axes7);
cla(handles.axes8);cla(handles.axes9);
UpdateFlowFretC;
handles.tryfitFR2 = 0;
guidata(hObject,handles);

function f_err = bindingmodel(params,CFPest,YFPest,Eeff)
KD = params(1);
EFRmax = params(2);

Ab = (CFPest+YFPest+KD - sqrt((CFPest+YFPest+KD).^2 - 4*CFPest.*YFPest))./(2*YFPest);
Eeffpred = Ab*EFRmax;

f_err = sum((Eeff-Eeffpred).^2);

function f_err = bindingmodel2(params,CFPest,YFPest,Eeff)
handles = guidata(gcf);
KD = params(1);
EFRmax = str2num(get(handles.EFRmax,'string'));

Ab = (CFPest+YFPest+KD - sqrt((CFPest+YFPest+KD).^2 - 4*CFPest.*YFPest))./(2*YFPest);
Eeffpred = Ab*EFRmax;

f_err = sum((Eeff-Eeffpred).^2);

function f_err = bindingmodel3(params,CFPest,YFPest,EFRETeff)
KD = params(1);
EFRETmax = params(2);

Db = (CFPest+YFPest+KD - sqrt((CFPest+YFPest+KD).^2 - 4*CFPest.*YFPest))./(2*CFPest);
EFRETeffpred = Db*EFRETmax;

f_err = sum((EFRETeff-EFRETeffpred).^2);

function f_err = bindingmodel4(params,CFPest,YFPest,EFRETeff)
handles = guidata(gcf);
KD = params(1);
EFRETmax = str2num(get(handles.EFRETmax,'string'));

Db = (CFPest+YFPest+KD - sqrt((CFPest+YFPest+KD).^2 - 4*CFPest.*YFPest))./(2*CFPest);
EFRETeffpred = Db*EFRETmax;

f_err = sum((EFRETeff-EFRETeffpred).^2);
%%


% --------------------------------------------------------------------
function printaxes_Callback(hObject, eventdata, handles)
% hObject    handle to printaxes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
h1 = gca;
h2 = figure;
copyobj(h1,h2);
Emax1 = str2num(get(handles.EFRmax,'string'));
Emax2 = str2num(get(handles.EFRETmax,'string'));
axisfontsize = 8;
fontname = 'Arial';
xlim = [5 1e5];
ylim = [0 1.1];
% yticks = [0 0.1 0.2 0.3 0.4 0.5];
yticks = [0 0.2 0.4 0.6 0.8 1.0];
% set(gca,'Units','Normalized','Position',[0.12 0.2 0.8 0.75],...
%     'color','white','FontSize',axisfontsize, 'FontName',fontname,...
%     'XLim',xlim,'YLim',ylim,'YTick',yticks);
set(gca,'Units','Normalized','Position',[0.12 0.2 0.8 0.75],...
    'color','white','FontSize',axisfontsize, 'FontName',fontname);
% l1 = findobj(gca,'type','line','marker','.');
% set(l1,'color',[0 0 0]);

% text(200,0.25,'EFb Y1875C','HorizontalAlignment','Left','FontName','Arial','FontSize',10);
set(findobj(gca,'type','line'),'MarkerSize',2);
set(findobj(gca,'type','text'),'FontName',fontname,'FontSize',axisfontsize);
FigW = 2*2.7;
FigH = 2*1.7;
if strcmpi(get(h1,'tag'),'axes7')
%     xlabel('CFP');
```

```
%     set(gca,'ylim',[0 1]);
%     xlabel('D_f_r_e_e (nM)');
%     hd = findobj(gca,'type','line');
%     for k=1:length(hd)
%         ydata = get(hd(k),'ydata');
%         ydata = ydata/Emax1;
%         set(hd(k),'ydata',ydata);
%     end
else
%     xlabel('A_f_r_e_e (nM)');
end
% ylabel('FRET efficiency');
set(h2, 'Units','Inches',...
                'Position',[3,3,FigW,FigH],...
                'Color','white');


% --------------------------------------------------------------------
function linlog_Callback(hObject, eventdata, handles)
% hObject    handle to linlog (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
linlog = get(gca,'XScale');
if strcmpi(linlog,'linear')
    set(gca,'XScale','log');
else
    set(gca,'XScale','linear');
end


% --------------------------------------------------------------------
function recalculate_Callback(hObject, eventdata, handles)
% hObject    handle to recalculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
listboxstr = get(handles.listbox,'string');
listboxval = listdlg('PromptString','Select constructs:', ...
    'SelectionMode','multiple','ListString',listboxstr);

% prompt = {'Change spurFR:','Change spurEFRET:'};
% dlg_title = 'Change spurFR or spurEFRET?';
% num_lines = 1;
% def = {'',''};
% answers = inputdlg(prompt,dlg_title,num_lines,def);
% spurFR = str2num(answers{1});
% spurEFRET = str2num(answers{2});
% if ~isempty(spurFR) | ~isempty(spurEFRET)
%     h = waitbar(0,'Please wait...');
%     for k=1:length(listboxval)
%         waitbar(k/length(listboxval),h);
%         structname = ['STR', num2str(listboxval(k),'%04.0f')];
%         DATA = load(handles.analysisfile,structname);
%         if ~isempty(spurFR)
%             DATA.(structname).spurFR = spurFR;
%         end
%         if ~isempty(spurEFRET)
%             DATA.(structname).spurEFRET = spurEFRET;
%         end
%
%         save(handles.analysisfile,'-struct','DATA','-append');
%     end
%     close(h);
%     msgbox('spurious values updated');
% end


choice = questdlg('Recalculate binding data?', ...
    'Recalculate', ...
    'Find KD, and Emax','Find KD, keep Emax','Cancel','Cancel');
switch choice
    case 'Find KD, and Emax'
        handles.tryfitFR1 = 1;
        handles.tryfitEFRET1 = 1;
    case 'Find KD, keep Emax'
        handles.tryfitFR2 = 1;
        handles.tryfitEFRET2 = 1;
    otherwise
        return;
end

h = waitbar(0,'Please wait...');
for k=1:length(listboxval)
    set(handles.listbox,'Value',listboxval(k));
    guidata(hObject,handles);
    listbox_Callback(handles.listbox,[],handles);
end
close(h);
msgbox('done');
```

```
% --------------------------------------------------------------------
function densityplot_Callback(hObject, eventdata, handles)
doth = findobj(gca,'type','line','Marker','.','tag','data');
xscale = get(gca,'XScale');
for k=1:length(doth)
    x = get(doth(k),'xdata')';
    y = get(doth(k),'ydata')';

%     xs = sort(x);
%     xmin = floor(xs(find(xs>0,1,'first')));
%     ymin = min(y);
%     xmax = min(xs(end),6e7);
%     ymax = min(max(y),1);
    xlims = get(gca,'xlim');
    ylims = get(gca,'ylim');
    xmin = xlims(1); xmax = xlims(2);
    ymin = ylims(1); ymax = ylims(2);

    if strcmp(xscale,'log')
        if xmin==0
            answer = inputdlg('Set XMin');
            xmin = str2num(answer{1});
        end
        dx = logspace(log10(xmin),log10(xmax),101);
    else
        dx = linspace(xmin,xmax,101);
    end
    dy = linspace(ymin,ymax,101);
    XYhist = zeros(length(dx)-1,length(dy)-1);
    X = zeros(length(dx)-1,length(dy)-1);
    Y = zeros(length(dx)-1,length(dy)-1);
    for m=1:length(dy)-1
        for j=1:length(dx)-1
            ind = find(x>dx(j) & x<dx(j+1) & y>dy(m) & y<dy(m+1));
            if strcmp(xscale,'log')
                area = (log10(dx(j+1))-log10(dx(j)))*(dy(m+1)-dy(m));
            else
                area = (dx(j+1)-dx(j))*(dy(m+1)-dy(m));
            end
            XYhist(j,m) = length(ind)/area;
            X(j,m) = mean(x(ind));
            Y(j,m) = mean(y(ind));
%             X(j,m) = dx(j);
%             Y(j,m) = dy(m);
        end
    end

    % divide XYhist into 20 colors
    bins = linspace((min(min(XYhist))*0.9), (1.1*max(max(XYhist))),21);

    map = jet;
    for m=1:length(bins)-1
        color = map(round(1+(m-1)/(length(bins)-1-1)*(length(map)-1)),:);
        ind = find(XYhist>bins(m) & XYhist<bins(m+1));
        line(X(ind),Y(ind),'linestyle','none','Marker','.','MarkerSize',5,'color',color);
    end

end
delete(doth);
set(gca,'children',flipud(get(gca,'children')));


% --------------------------------------------------------------------
function ErrVSKD_Callback(hObject, eventdata, handles)
% hObject    handle to ErrVSKD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

UpdateFlowFretC(2);
% guidata(hObject,handles);


% --------------------------------------------------------------------
function multiselect_Callback(hObject, eventdata, handles)
% hObject    handle to multiselect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

listboxstr = get(handles.listbox,'string');

[choice,v] = listdlg('PromptString','Choose files:',...
                'SelectionMode','Multiple',...
                'ListString',listboxstr);
if v
    set(handles.listbox,'Value',choice);
    listbox_Callback(handles.listbox,[],handles);
    guidata(hObject,handles);
end
```

```
function [CFPd, YFPd, YFPf, Cherry] = Fdecomp(params,cfp,yfp,fret,BL3);

% params = [RD1,RD2,RD3,RA,RA2,RC1,RC2,RC3];
RD1 = params(1);
RD2 = params(2);
RD3 = params(3);
RA = params(4);
RA2 = params(5);
RC1 = params(6);
RC2 = params(7);
RC3 = params(8);

% M = [1/RD1          0            0            1/RC1 ; ...
%      1              1            1              1  ; ...
%      RD2/RD1       1/RA          0           RC2/RC1; ...
%      RD3/RD1       1/RA2        0           RC2*RC3/RC1];

% M = [1/RD1          0            0            1/(RC2*RC3) ; ...
%      1              1            1         RC1/(RC2*RC3)  ; ...
%      RD2/RD1       1/RA          0            1/RC3; ...
%      RD3/RD1       1/RA2        0               1];

% M = [1/RD1          0            0               0 ; ...
%      1              1            1               0  ; ...
%      RD2/RD1       1/RA          0            1/RC3; ...
%      RD3/RD1       1/RA2        0               1];

M = [1/RD1          0            0               0 ; ...
     1              1            1               0  ; ...
     RD2/RD1       1/RA          0               0; ...
     RD3/RD1       1/RA2        0               1];

% y = Mx --> x = M\y
% check whether row/column vectors
if size(cfp,2)==1
    cfp = cfp';
end
if size(fret,2)==1
    fret = fret';
end
if size(yfp,2)==1
    yfp = yfp';
end
if size(BL3,2)==1
    BL3 = BL3';
end
solns = M\[cfp ; fret ; yfp ; BL3];
CFPd = solns(1,:)';
YFPd = solns(2,:)';
YFPf = solns(3,:)';
Cherry = solns(4,:)';
```

References

1. Mori, Y., et al., *Primary structure and functional expression from complementary DNA of a brain calcium channel.* Nature, 1991. **350**(6317): p. 398-402.
2. Hillman, D., et al., *Localization of P-type calcium channels in the central nervous system.* Proc Natl Acad Sci U S A, 1991. **88**(16): p. 7076-80.
3. Dunlap, K., J.I. Luebke, and T.J. Turner, *Exocytotic Ca2+ channels in mammalian central neurons.* Trends Neurosci, 1995. **18**(2): p. 89-98.
4. Westenbroek, R.E., et al., *Immunochemical identification and subcellular distribution of the alpha 1A subunits of brain calcium channels.* J Neurosci, 1995. **15**(10): p. 6403-18.
5. Craig, P.J., et al., *Distribution of the voltage-dependent calcium channel alpha(1A) subunit throughout the mature rat brain and its relationship to neurotransmitter pathways.* J Comp Neurol, 1998. **397**(2): p. 251-67.
6. Regan, L.J., *Voltage-dependent calcium currents in Purkinje cells from rat cerebellar vermis.* J Neurosci, 1991. **11**(7): p. 2259-69.
7. Usowicz, M.M., et al., *P-type calcium channels in the somata and dendrites of adult cerebellar Purkinje cells.* Neuron, 1992. **9**(6): p. 1185-99.
8. Indriati, D.W., et al., *Quantitative localization of Cav2.1 (P/Q-type) voltage-dependent calcium channels in Purkinje cells: somatodendritic gradient and distinct somatic coclustering with calcium-activated potassium channels.* J Neurosci, 2013. **33**(8): p. 3668-78.
9. Uchitel, O.D., et al., *P-type voltage-dependent calcium channel mediates presynaptic calcium influx and transmitter release in mammalian synapses.* Proc Natl Acad Sci U S A, 1992. **89**(8): p. 3330-3.
10. Zucker, R.S. and W.G. Regehr, *Short-term synaptic plasticity.* Annu Rev Physiol, 2002. **64**: p. 355-405.
11. Finch, E.A., K. Tanaka, and G.J. Augustine, *Calcium as a trigger for cerebellar long-term synaptic depression.* Cerebellum, 2012. **11**(3): p. 706-17.
12. Lamont, M.G. and J.T. Weber, *The role of calcium in synaptic plasticity and motor learning in the cerebellar cortex.* Neurosci Biobehav Rev, 2012. **36**(4): p. 1153-62.
13. Wheeler, D.G., et al., *Ca(V)1 and Ca(V)2 channels engage distinct modes of Ca(2+) signaling to control CREB-dependent gene expression.* Cell, 2012. **149**(5): p. 1112-24.
14. Miyazaki, T., et al., *Cav2.1 in cerebellar Purkinje cells regulates competitive excitatory synaptic wiring, cell survival, and cerebellar biochemical compartmentalization.* J Neurosci, 2012. **32**(4): p. 1311-28.
15. Womack, M.D. and K. Khodakhah, *Dendritic control of spontaneous bursting in cerebellar Purkinje cells.* J Neurosci, 2004. **24**(14): p. 3511-21.
16. Womack, M. and K. Khodakhah, *Active contribution of dendrites to the tonic and trimodal patterns of activity in cerebellar Purkinje neurons.* J Neurosci, 2002. **22**(24): p. 10603-12.

17. Charlton, M.P., S.J. Smith, and R.S. Zucker, *Role of presynaptic calcium ions and channels in synaptic facilitation and depression at the squid giant synapse.* J Physiol, 1982. **323**: p. 173-93.
18. Regehr, W.G., *Short-term presynaptic plasticity.* Cold Spring Harb Perspect Biol, 2012. **4**(7): p. a005702.
19. Abbott, L.F. and W.G. Regehr, *Synaptic computation.* Nature, 2004. **431**(7010): p. 796-803.
20. De Waard, M. and K.P. Campbell, *Subunit regulation of the neuronal alpha 1A Ca2+ channel expressed in Xenopus oocytes.* J Physiol, 1995. **485 (Pt 3)**: p. 619-34.
21. Patil, P.G., D.L. Brody, and D.T. Yue, *Preferential closed-state inactivation of neuronal calcium channels.* Neuron, 1998. **20**(5): p. 1027-38.
22. DeMaria, C.D., et al., *Calmodulin bifurcates the local Ca2+ signal that modulates P/Q-type Ca2+ channels.* Nature, 2001. **411**(6836): p. 484-9.
23. Lee, A., T. Scheuer, and W.A. Catterall, *Ca2+/calmodulin-dependent facilitation and inactivation of P/Q-type Ca2+ channels.* J Neurosci, 2000. **20**(18): p. 6830-8.
24. Erickson, M.G., et al., *Preassociation of calmodulin with voltage-gated Ca(2+) channels revealed by FRET in single living cells.* Neuron, 2001. **31**(6): p. 973-85.
25. Adams, P.J., et al., *Contribution of calcium-dependent facilitation to synaptic plasticity revealed by migraine mutations in the P/Q-type calcium channel.* Proc Natl Acad Sci U S A, 2010. **107**(43): p. 18694-9.
26. Borst, J.G. and B. Sakmann, *Facilitation of presynaptic calcium currents in the rat brainstem.* J Physiol, 1998. **513 (Pt 1)**: p. 149-55.
27. Catterall, W.A., K. Leal, and E. Nanou, *Calcium channels and short-term synaptic plasticity.* J Biol Chem, 2013. **288**(15): p. 10742-9.
28. Mochida, S., et al., *Regulation of presynaptic Ca(V)2.1 channels by Ca2+ sensor proteins mediates short-term synaptic plasticity.* Neuron, 2008. **57**(2): p. 210-6.
29. Xu, J., L. He, and L.G. Wu, *Role of Ca(2+) channels in short-term synaptic plasticity.* Curr Opin Neurobiol, 2007. **17**(3): p. 352-9.
30. Lee, A., et al., *Molecular determinants of Ca(2+)/calmodulin-dependent regulation of Ca(v)2.1 channels.* Proc Natl Acad Sci U S A, 2003. **100**(26): p. 16059-64.
31. Mori, M.X., et al., *Crystal structure of the CaV2 IQ domain in complex with Ca2+/calmodulin: high-resolution mechanistic implications for channel regulation by Ca2+.* Structure, 2008. **16**(4): p. 607-20.
32. Kim, E.Y., et al., *Structures of CaV2 Ca2+/CaM-IQ domain complexes reveal binding modes that underlie calcium-dependent inactivation and facilitation.* Structure, 2008. **16**(10): p. 1455-67.
33. Soong, T.W., et al., *Systematic identification of splice variants in human P/Q-type channel alpha1(2.1) subunits: implications for current density and Ca2+-dependent inactivation.* J Neurosci, 2002. **22**(23): p. 10142-52.
34. Tay, L.H., et al., *Nanodomain Ca(2)(+) of Ca(2)(+) channels detected by a tethered genetically encoded Ca(2)(+) sensor.* Nat Commun, 2012. **3**: p. 778.

35. Chaudhuri, D., J.B. Issa, and D.T. Yue, *Elementary mechanisms producing facilitation of Cav2.1 (P/Q-type) channels.* J Gen Physiol, 2007. **129**(5): p. 385-401.
36. Tadross, M.R., R.W. Tsien, and D.T. Yue, *Ca2+ channel nanodomains boost local Ca2+ amplitude.* Proc Natl Acad Sci U S A, 2013. **110**(39): p. 15794-9.
37. Ben-Johny, M., et al., *Conservation of Ca2+/calmodulin regulation across Na and Ca2+ channels.* Cell, 2014. **157**(7): p. 1657-70.
38. Ho, S.N., et al., *Site-directed mutagenesis by overlap extension using the polymerase chain reaction.* Gene, 1989. **77**(1): p. 51-9.
39. Brody, D.L., et al., *Bursts of action potential waveforms relieve G-protein inhibition of recombinant P/Q-type Ca2+ channels in HEK 293 cells.* J Physiol, 1997. **499 (Pt 3)**: p. 637-44.
40. McCleskey, E.W. and W. Almers, *The Ca channel in skeletal muscle is a large pore.* Proc Natl Acad Sci U S A, 1985. **82**(20): p. 7149-53.
41. Hess, P., J.B. Lansman, and R.W. Tsien, *Calcium channel selectivity for divalent and monovalent cations. Voltage and concentration dependence of single channel current in ventricular heart cells.* J Gen Physiol, 1986. **88**(3): p. 293-319.
42. Yang, J., et al., *Molecular determinants of Ca2+ selectivity and ion permeation in L-type Ca2+ channels.* Nature, 1993. **366**(6451): p. 158-61.
43. Ellinor, P.T., et al., *Ca2+ channel selectivity at a single locus for high-affinity Ca2+ interactions.* Neuron, 1995. **15**(5): p. 1121-32.
44. Chaudhuri, D., et al., *Alternative splicing as a molecular switch for Ca2+/calmodulin-dependent facilitation of P/Q-type Ca2+ channels.* J Neurosci, 2004. **24**(28): p. 6334-42.
45. Brehm, P. and R. Eckert, *Calcium entry leads to inactivation of calcium channel in Paramecium.* Science, 1978. **202**: p. 1203-1206.
46. Chao, S.H., et al., *Activation of calmodulin by various metal cations as a function of ionic radius.* Mol. Pharmacol., 1984. **26**(1): p. 75-82.
47. Chaudhuri, D., et al., *Developmental activation of calmodulin-dependent facilitation of cerebellar P-type Ca2+ current.* J Neurosci, 2005. **25**(36): p. 8282-94.
48. Lee, A., et al., *Ca2+/calmodulin binds to and modulates P/Q-type calcium channels.* Nature, 1999. **399**(6732): p. 155-9.
49. Tadross, M.R., I.E. Dick, and D.T. Yue, *Mechanism of local and global Ca2+ sensing by calmodulin in complex with a Ca2+ channel.* Cell, 2008. **133**(7): p. 1228-40.
50. Mintz, I.M., B.L. Sabatini, and W.G. Regehr, *Calcium control of transmitter release at a cerebellar synapse.* Neuron, 1995. **15**(3): p. 675-88.
51. Chang, S.Y., et al., *Age and gender-dependent alternative splicing of P/Q-type calcium channel EF-hand.* Neuroscience, 2007. **145**(3): p. 1026-36.
52. Ben-Johny, M., et al., *Dynamic switching of calmodulin interactions underlies Ca2+ regulation of CaV1.3 channels.* Nat Commun, 2013. **4**: p. 1717.
53. Tazi, J., N. Bakkour, and S. Stamm, *Alternative splicing and disease.* Biochim Biophys Acta, 2009. **1792**(1): p. 14-26.

54. Peterson, B.Z., et al., *Calmodulin is the Ca2+ sensor for Ca2+ -dependent inactivation of L-type calcium channels.* Neuron, 1999. **22**(3): p. 549-58.
55. Yang, P.S., et al., *Switching of Ca2+-dependent inactivation of Ca(v)1.3 channels by calcium binding proteins of auditory hair cells.* J Neurosci, 2006. **26**(42): p. 10677-89.
56. Tanaka, K., et al., *Ca2+ requirements for cerebellar long-term synaptic depression: role for a postsynaptic leaky integrator.* Neuron, 2007. **54**(5): p. 787-800.
57. Lin, K.H., E. Erazo-Fischer, and H. Taschenberger, *Similar intracellular Ca2+ requirements for inactivation and facilitation of voltage-gated Ca2+ channels in a glutamatergic mammalian nerve terminal.* J Neurosci, 2012. **32**(4): p. 1261-72.
58. Wu, L.G., J.G. Borst, and B. Sakmann, *R-type Ca2+ currents evoke transmitter release at a rat central synapse.* Proc Natl Acad Sci U S A, 1998. **95**(8): p. 4720-5.
59. Wu, L.G., et al., *Calcium channel types with distinct presynaptic localization couple differentially to transmitter release in single calyx-type synapses.* J Neurosci, 1999. **19**(2): p. 726-36.
60. Iwasaki, S. and T. Takahashi, *Developmental changes in calcium channel types mediating synaptic transmission in rat auditory brainstem.* J Physiol, 1998. **509 (Pt 2)**: p. 419-23.
61. Gruol, D., M. Manto, and D. Haines, *Ca2+ signaling in cerebellar Purkinje neurons--editorial.* Cerebellum, 2012. **11**(3): p. 605-8.
62. Hartmann, J. and A. Konnerth, *Determinants of postsynaptic Ca2+ signaling in Purkinje neurons.* Cell Calcium, 2005. **37**(5): p. 459-66.
63. Otis, T.S. and J.C. Jen, *Blessed are the pacemakers.* Nat Neurosci, 2006. **9**(3): p. 297-8.
64. Walter, J.T., et al., *Decreases in the precision of Purkinje cell pacemaking cause cerebellar dysfunction and ataxia.* Nat Neurosci, 2006. **9**(3): p. 389-97.
65. Hoebeek, F.E., et al., *Increased noise level of purkinje cell activities minimizes impact of their modulation during sensorimotor control.* Neuron, 2005. **45**(6): p. 953-65.
66. Medina, J.F. and K. Khodakhah, *Neuroscience: Spikes timed through inhibition.* Nature, 2012. **481**(7382): p. 446-7.
67. Person, A.L. and I.M. Raman, *Purkinje neuron synchrony elicits time-locked spiking in the cerebellar nuclei.* Nature, 2012. **481**(7382): p. 502-5.
68. Person, A.L. and I.M. Raman, *Synchrony and neural coding in cerebellar circuits.* Front Neural Circuits, 2012. **6**: p. 97.
69. Swensen, A.M. and B.P. Bean, *Ionic mechanisms of burst firing in dissociated Purkinje neurons.* J Neurosci, 2003. **23**(29): p. 9650-63.
70. Womack, M.D., C. Chevez, and K. Khodakhah, *Calcium-activated potassium channels are selectively coupled to P/Q-type calcium channels in cerebellar Purkinje neurons.* J Neurosci, 2004. **24**(40): p. 8818-22.

71. Arata, A. and M. Ito, *Synaptic transmission and long-term depression in Purkinje cells in an in vitro block preparation of the cerebellum isolated from neonatal rats.* Prog Brain Res, 2005. **148**: p. 111-23.
72. Wheeler, D.B., A. Randall, and R.W. Tsien, *Roles of N-type and Q-type Ca2+ channels in supporting hippocampal synaptic transmission.* Science, 1994. **264**(5155): p. 107-11.
73. Cuttle, M.F., et al., *Facilitation of the presynaptic calcium current at an auditory synapse in rat brainstem.* J Physiol, 1998. **512 (Pt 3)**: p. 723-9.
74. Schneggenburger, R. and E. Neher, *Intracellular calcium dependence of transmitter release rates at a fast central synapse.* Nature, 2000. **406**(6798): p. 889-93.
75. Muller, M., F. Felmy, and R. Schneggenburger, *A limited contribution of Ca2+ current facilitation to paired-pulse facilitation of transmitter release at the rat calyx of Held.* J Physiol, 2008. **586**(Pt 22): p. 5503-20.
76. Sheng, J., et al., *Calcium-channel number critically influences synaptic strength and plasticity at the active zone.* Nat Neurosci, 2012. **15**(7): p. 998-1006.
77. Regehr, W.G., K.R. Delaney, and D.W. Tank, *The role of presynaptic calcium in short-term enhancement at the hippocampal mossy fiber synapse.* J Neurosci, 1994. **14**(2): p. 523-37.
78. Awatramani, G.B., G.D. Price, and L.O. Trussell, *Modulation of transmitter release by presynaptic resting potential and background calcium levels.* Neuron, 2005. **48**(1): p. 109-21.
79. Inchauspe, C.G., I.D. Forsythe, and O.D. Uchitel, *Changes in synaptic transmission properties due to the expression of N-type calcium channels at the calyx of Held synapse of mice lacking P/Q-type calcium channels.* J Physiol, 2007. **584**(Pt 3): p. 835-51.
80. Robinson, C.V., A. Sali, and W. Baumeister, *The molecular sociology of the cell.* Nature, 2007. **450**(7172): p. 973-82.
81. Bruckner, A., et al., *Yeast two-hybrid, a powerful tool for systems biology.* Int J Mol Sci, 2009. **10**(6): p. 2763-88.
82. Kenworthy, A.K., *Imaging protein-protein interactions using fluorescence resonance energy transfer microscopy.* Methods, 2001. **24**(3): p. 289-96.
83. Erickson, M.G., et al., *FRET two-hybrid mapping reveals function and location of L-type Ca2+ channel CaM preassociation.* Neuron, 2003. **39**(1): p. 97-107.
84. Chen, H., H.L. Puhl, 3rd, and S.R. Ikeda, *Estimating protein-protein interaction affinity in living cells using quantitative Forster resonance energy transfer measurements.* J Biomed Opt, 2007. **12**(5): p. 054011.
85. Rizzo, M.A., et al., *An improved cyan fluorescent protein variant useful for FRET.* Nat Biotechnol, 2004. **22**(4): p. 445-9.
86. Nagai, T., et al., *A variant of yellow fluorescent protein with fast and efficient maturation for cell-biological applications.* Nat Biotechnol, 2002. **20**(1): p. 87-90.
87. Van Munster, E.B., et al., *Fluorescence resonance energy transfer (FRET) measurement by gradual acceptor photobleaching.* J Microsc, 2005. **218**(Pt 3): p. 253-62.

88. Pinsky, B.G., et al., *Phase-resolved fluorescence lifetime measurements for flow cytometry.* Cytometry, 1993. **14**(2): p. 123-35.
89. Sun, Y. and A. Periasamy, *Localizing protein-protein interactions in living cells using fluorescence lifetime imaging microscopy.* Methods Mol Biol, 2015. **1251**: p. 83-107.
90. Leavesley, S.J., et al., *Assessing FRET using spectral techniques.* Cytometry A, 2013. **83**(10): p. 898-912.
91. Szaloki, N., et al., *High throughput FRET analysis of protein-protein interactions by slide-based imaging laser scanning cytometry.* Cytometry A, 2013. **83**(9): p. 818-29.
92. You, X., et al., *Intracellular protein interaction mapping with FRET hybrids.* Proc Natl Acad Sci U S A, 2006. **103**(49): p. 18458-63.
93. Banning, C., et al., *A flow cytometry-based FRET assay to identify and analyse protein-protein interactions in living cells.* PLoS One, 2010. **5**(2): p. e9344.
94. Berney, C. and G. Danuser, *FRET or no FRET: a quantitative comparison.* Biophys J, 2003. **84**(6): p. 3992-4010.
95. Zacharias, D.A., et al., *Partitioning of lipid-modified monomeric GFPs into membrane microdomains of live cells.* Science, 2002. **296**(5569): p. 913-6.
96. Seybold, P.G., M. Gouterman, and J. Callis, *Calorimetric, photometric and lifetime determinations of fluorescence yields of fluorescein dyes.* Photochem Photobiol, 1969. **9**(3): p. 229-42.
97. Shaner, N.C., P.A. Steinbach, and R.Y. Tsien, *A guide to choosing fluorescent proteins.* Nat Methods, 2005. **2**(12): p. 905-9.
98. Chen, H., et al., *Measurement of FRET efficiency and ratio of donor to acceptor concentration in living cells.* Biophys J, 2006. **91**(5): p. L39-41.
99. Nagy, P., et al., *Novel calibration method for flow cytometric fluorescence resonance energy transfer measurements between visible fluorescent proteins.* Cytometry A, 2005. **67**(2): p. 86-96.
100. Harkiolaki, M., et al., *Structural basis for SH3 domain-mediated high-affinity binding between Mona/Gads and SLP-76.* EMBO J, 2003. **22**(11): p. 2571-82.
101. Beuschlein, F., et al., *Constitutive activation of PKA catalytic subunit in adrenal Cushing's syndrome.* N Engl J Med, 2014. **370**(11): p. 1019-28.
102. Cao, Y., et al., *Activating Hotspot L205R Mutation in PRKACA and Adrenal Cushing's Syndrome.* Science, 2014.
103. Di Dalmazi, G., et al., *Novel somatic mutations in the catalytic subunit of the protein kinase A as a cause of adrenal Cushing's syndrome: a European multicentric study.* J Clin Endocrinol Metab, 2014. **99**(10): p. E2093-100.
104. Goh, G., et al., *Recurrent activating mutation in PRKACA in cortisol-producing adrenal tumors.* Nat Genet, 2014.
105. Sato, Y., et al., *Recurrent somatic mutations underlie corticotropin-independent Cushing's syndrome.* Science, 2014. **344**(6186): p. 917-20.
106. Taylor, S.S., et al., *PKA: lessons learned after twenty years.* Biochim Biophys Acta, 2013. **1834**(7): p. 1271-8.

107. Cheng, X., C. Phelps, and S.S. Taylor, *Differential binding of cAMP-dependent protein kinase regulatory subunit isoforms Ialpha and IIbeta to the catalytic subunit.* J Biol Chem, 2001. **276**(6): p. 4102-8.
108. Scott, J.D., et al., *Identification of an inhibitory region of the heat-stable protein inhibitor of the cAMP-dependent protein kinase.* Proc Natl Acad Sci U S A, 1985. **82**(13): p. 4379-83.
109. Depry, C., M.D. Allen, and J. Zhang, *Visualization of PKA activity in plasma membrane microdomains.* Mol Biosyst, 2011. **7**(1): p. 52-8.
110. Moore, M.J., J.A. Adams, and S.S. Taylor, *Structural basis for peptide binding in protein kinase A. Role of glutamic acid 203 and tyrosine 204 in the peptide-positioning loop.* J Biol Chem, 2003. **278**(12): p. 10613-8.
111. Pepperkok, R. and J. Ellenberg, *High-throughput fluorescence microscopy for systems biology.* Nat Rev Mol Cell Biol, 2006. **7**(9): p. 690-6.
112. Cheung, J., et al., *Structural insights into mis-regulation of protein kinase A in human tumors.* Proc Natl Acad Sci U S A, 2015. **112**(5): p. 1374-9.
113. Calebiro, D., et al., *PKA catalytic subunit mutations in adrenocortical Cushing's adenoma impair association with the regulatory subunit.* Nat Commun, 2014. **5**: p. 5680.

Curriculum Vitae

Shin Rong Lee 3/1/2016

Educational History:

M.D. expected	2016	School of Medicine	Johns Hopkins University
Ph.D. expected	2016	Program in Biomedical Engineering Mentor: David T. Yue, M.D., Ph.D.	Johns Hopkins University
B.S.	2007	Electrical Engineering	Yale University

Other Professional Experience:

Research rotation	2008	Lab of Reza Shadmehr, Ph.D.	Johns Hopkins University

Fellowships:

2007-2009, 2014-2016	Medical scientist training program	NIH

Awards:

2007	Phi Beta Kappa Honor Society	Yale University
2007	Edward O. Lanphier Memorial Prize in Electrical Engineering	Yale University
2006	Tau Beta Pi Engineering Honor Society	Yale University
2006	Belle and Carl Morse Junior Prize in Electrical Engineering	Yale University
2006	Yale College Dean's Research Fellowship	Yale University

Publications:

Lee, S.R., Cooper, M., Eckhauser, F.E., Weiss, M.J. *et al.* (2016) Pancreatic metastasectomies at Johns Hopkins Hospital – a retrospective review. *Under preparation.*

Lee, S.R., Sang, L.J., and Yue, D.T. (2016) Uncovering aberrant mutant PKA function with flow cytometric FRET. *Cell Reports*, In press

Ben-Johny, M., Dick, I.E., Sang, L., Limpitikul, W.B., Kang, P.W., Niu, J., Banerjee, R., Yang, W., Babich, J.S., Issa, J.B., **Lee, S.R.**, Namkung, H., *et al*.. (2015) Towards a Unified Theory of Calmodulin Regulation (Calmodulation) of Voltage-Gated Calcium and Sodium Channels. *Current molecular pharmacology 8*, 188-205.

Dick, I.E., Limpitikul, W.B., Niu, J., Banerjee, R., Issa, J.B., Ben-Johny, M., Adams, P.J., Kang, P.W., **Lee, S.R.**, Sang, L.*, et al.*. (2015) A rendezvous with the queen of ion channels: Three decades of ion channel research by David T. Yue and his Calcium Signals Laboratory. *Channels.* doi*:* 19336950.2015.105127.

Lee, S.R., Adams, P.J., and Yue, D.T. Large Ca^{2+}-dependent facilitation of $Ca_V2.1$ channels revealed by Ca^{2+} photouncaging. (2015) *The Journal of Physiology*, **593**, 2753-2778.

Park, S.A., **Lee, S.R.**, Tung, L. and Yue, D.T. (2014) Optical mapping of optogenetically shaped cardiac action potentials. *Scientific Reports* 4, 6125.

Posters and abstracts:

Lee, S.R., Sang, L.J., and Yue, D.T. Quantitative FRET with flow cytometry—uncovering aberrant mutant PKA function via FRET-based binding assays and activity sensors. (2015) *Johns Hopkins MD-PhD Annual Program Retreat* [Podium presentation].

Lee, S.R., Sang, L.J., and Yue, D.T. (2014) High-throughput FRET-based binding assays with flow cytometry. *Biophysical Journal* 106, 245a [Podium presentation].

Lee, S.R., Ben Johny, M., and Yue, D.T. (2013) Large Ca^{2+}-dependent facilitation of CaV2.1 channels induced by Ca^{2+} photouncaging (abstr.). *Biophysical Journal* 104, 461a.

Park, S.A., **Lee, S.R.**, Tung, L, Yue, D.T. (2012) Optical mapping of optogenetically shaped cardiac action potentials. *Biophysical Journal* 102.

Service and leadership:

2011-2014 Mentor. Calcium Signals Lab. Johns Hopkins University

Trained junior graduate students and new postdoctoral fellows. Taught molecular biology techniques, electrophysiology, Ca^{2+} imaging and uncaging, confocal fluorescence imaging, and flow cytometry.

2012 Student Advisor. Calcium Signals Lab. Johns Hopkins University.

Mentored a high school student during his summer project. Taught molecular biology, fluorescence imaging techniques and MATLAB coding. Designed experiments and modified microscopes to 1) allow Ca^{2+} imaging and uncaging using esterified Ca^{2+} dyes and cages, and 2) assess binding between CaMKII subunits with FRET.

2012 Teaching Assistant. EN.580.420: Build-A-Genome.

Taught and assisted students with molecular biology techniques during lab sessions three times a week. Helped students troubleshoot problems with PCR and cloning, and was responsible for setup and maintenance of lab equipment and reagents

2011 Teaching assistant. EN.580.421: Systems Bioengineering I

Taught weekly sections covering topics in membrane biophysics. Graded assignments and examinations. One-on-one tutoring during office hours.

2010	Inventor. Dept. of Biomedical Engineering
	Led team in designing low-cost blood pressure monitoring device for use in developing countries. Presented and demonstrated device's use and features at the non-profit health organization JHPIEGO.
2009	Volunteer. Health Leads, Baltimore MD
	Helped uninsured patients at the Hopkins Bayview Medical Center's emergency department enroll in medical assistance programs.
2008	Webmaster and co-organizer. Revisit Weekend, Johns Hopkins SOM
	Created website for admitted students to register for Revisit Weekend events and learn more about Johns Hopkins, as well as assisted with organization of the event.
2007-2008	Co-webmaster. Johns Hopkins MD-PhD Program
	Redesigned and created new content for the JHU MD-PhD website
2006-2007	Secretary. Yale IEEE
	Organized social events within the engineering community as well as neighborhood events to stimulate interest in electrical engineering.
2006-2007	Computing assistant. Yale College
	Assisted students with a variety of IT-related problems, from networking support to virus/malware removal, data recovery and computer repairs.
2006	Tutor. EENG 228: Intro to Signals and Systems
	Tutored 2 students weekly, reviewing lecture materials and providing guidance with problem sets.
2006	Peer tutor. EENG 227 and EENG 229: Circuits and Design
	Helped students in this laboratory course troubleshoot their electrical circuits, explained course material and advised them on their term projects.