METHODS FOR ROBUST MEASUREMENTS IN LOW SIGNAL-TO-NOISE SYSTEMS AND THEIR APPLICATION TO NEMATODE MECHANORECEPTORS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Joy A. Franco

August 2021

This dissertation is online at: https://purl.stanford.edu/hy400rj1686

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Miriam Goodman, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Thomas Kenny, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ovijit Chaudhuri

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Beth Pruitt

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Animals small and large, vertebrate and invertebrate alike, use specialized sensory cells to detect mechanical cues in their environment. These mechanoreceptors are tuned to specific stimulus profiles and often work as part of an ensemble of sensory cells that allows animals to detect multiple types of mechanical stimuli in concert. Experimental data and theoretical modeling predicts that mechanoreceptor function and form are tightly coupled, however testing this hypothesis has faced numerous technical challenges. These biological sensors are typically embedded in tissue that limits direct access to the fundamental sensing units (i.e., the Mechanoelectrical Transduction, MeT, ion channels).

The nematode *Caenorhabditis elegans* is uniquely poised as an ideal system for probing the relationship between receptor form and function. Prior work has established the physiological relevance of *C. elegans* receptor shape, yet the genetic and molecular interactions leading to this relevance has not been elucidated. This is likely again due to technical limitations and a lack of tools available for (1) controlling receptor morphology and (2) reliably scoring morphology in various genetic backgrounds. In this thesis, by applying mechanical engineering principles to this biological system, I have developed novel experimental and analytical approaches for overcoming this technical gap. In early work I refined a previously established yet underutilized method for studying *C. elegans* mechanoreceptors *in vitro*. Using this approach I have demonstrated that mechanoreceptor shape can be synthetically controlled via contactless micropatterning of peanut lectin.

To achieve a robust analysis of these experimental results I developed a suite of image processing code to conduct both automated and supervised morphological measurements. To validate these approaches I applied these methods to quantifying the effects of various genetic mutations to

MeT channel localization and mechanoreceptor morphology *in vitro*. These mechanoreceptors often show significantly greater morphological heterogeneity than their *in vivo* correlates, effectively raising the noise floor and making it difficult to derive meaningful measurements. This decrease in signal-to-noise requires experimenters to exercise subjectivity when choosing representative samples and reduces confidence in drawing conclusions from these measurements. To circumvent this challenge I show that micropatterning can be used *in vitro* to reduce biological variance, and that image segmentation can be used *in silico* to reduce operator bias, increase robustness and reproducibility. Combined, these approaches allow for reliable measurements in otherwise noisy backgrounds.

This thesis has generated a number of tools that will enable researchers to conduct robust morphological measurements in low signal-to-noise biological systems such as nematode mechanoreceptors. This work establishes patterning tools that can be used to generate controlled variations in neurite curvature in the future and further probe the relationship between ion channel localization, receptor shape, and mechanosensory function. The processes developed in this work can also be easily extended to cryo-electron tomography, atomic force microscopy, and real time imaging of organelle trafficking.

Acknowledgements

More than ten years ago I returned to community college, not only to eventually transfer to a four-year university to attain a Bachelor's degree, but to ultimately pursue a PhD. I returned to college because I knew I wanted a research career. I wanted to chase big problems and dedicate my life to the pursuit of knowledge. By that point in my life I had met a number of fascinating people from racing bicycles in the Bay Area who had exposed me to an academic world that seemed completely out of reach for someone like me. But they believed in me. They saw my inner nerd shining through and they were committed to helping me fulfill that glimmer of potential. Though this set the stage for my return to education, it became a sort of theme from Cañada College to San José State to Stanford. Along the way countless people have contributed to this dream and helped to make it possible. I will never be able to fully express the tremendous amount of gratitude I feel towards each of these people, nor would it be possible to thank every single one of them, but I can at least give it a go.

This body of work and my PhD journey in general would not have been possible without my PhD advisors, Miriam Goodman and Beth Pruitt. Each of them committed to supporting me when I was at my absolute worst and most would have passed me over for a more promising candidate. Each of them kept me from leaving my PhD journey when it felt impossible to continue. Beth introduced me to a world of mechanobiology and biofabrication methods. In her lab I learned to engineer microsystems for probing single cells. This thesis would not have been possible without the techniques and insight I gained from her and the teams she has led. When I met Miriam I was completely lost in my PhD program. I felt so disconnected from the experience I had dreamt of. For whatever reason, she took a risk on me, welcoming me with open arms into her lab. It was exactly

where I needed to be. In the past few years she has been a constant pillar of support in science and beyond, providing a source of strength and calm in the ultimate storm. This thesis would never have happened without her. I am so grateful for both of my mentors, and I am proud to be their student.

At the risk of sounding cliché, the words in this thesis are only the tip of the iceberg. They are built on a foundation of knowledge, experiences, and contributions from others that reach far beyond these pages. I can only hope to remember each of them. I thank Alakananda Das for contributing her expertise in genetics, molecular biology, and microscopy to this work. It has been a tremendous pleasure to learn from you these past few years, Alka. None of us would have hoped for the pandemic, but I am glad that we were able to find a collaboration in it. I thank Carmen Liao for her patience with me in the lab, teaching me to work with worms, performing injections, supporting experiments...it's a very long list. Thank you Carmen. I thank Chensong Zhang and Yiorgo Skiniotis for their contributions to the cryoET as well as the Fantner Lab at EPFL and Marcin Walkiewicz for theirs to the bioAFM in Chapter 2. I thank the staff at the Stanford Nanofabrication Facilities, Swaroop Kommera, Mary Tang, and Carsen Kline, for their support with protein micropatterning experiments over many years.

I extend my deepest gratitude to the ECM team, without whom the entire fourth chapter of this thesis would not have been possible: Dail Chapman, Lucy Wang, and Lingxin Wang. I also thank Ehsan Razei whose work in the lab has helped to further our efforts to translate our scientific findings into a working mechanical model of sensory transduction.

My studies as a PhD student have been very generously supported by numerous awards for which I am extremely grateful. The NSF GRFP provided three years of support, the NIH DSPAN F99 award has helped me in the last two years of my time as a student, and the School of Engineering and Department of Mechanical Engineering help to supplement. A VPGE EDGE award helped to make sure that I was able to purchase the technology I needed to perform my quantitative work. The DSPAN program has also provided me a source of peer support that has evolved into a remote peer mentorship group that has been essential to my successful transition into a postdoc. I thank Ubadah Sabbagh, Kaela Singleton, and Cory White for their mentorship, kindness, and generosity of spirit. I promise to always pay this forward.

Throughout my time as a PhD student I have been extremely lucky to have had multiple lab

mates who were in a similar position as me: an engineering student struggling with their love-hate relationship of interdisciplinary research. My peer mentor, Adam Nekimken, truly helped me develop as a scientist-engineer. He helped me grow intellectually and for that I am eternally grateful. His work contributed to how I think about neurite mechanics in the worm and that has been of critical value to my dissertation progress.

Along those lines, I thank my lab mates Sammata Katta and Sylvia Fechner for teaching me about all things ion channels. You sparked a love for MEC-4 in me that I honestly don't think will ever die. I can't tell you enough how much our science conversations meant to me. They were everything. They were literally the inspiration for everything about this thesis. By sharing your work with me you made me fascinated with this ion channel and determined to understand it to the fullest. I'm so lucky to know you both.

The biofabrication processes developed in this were truly inspired by conversations with many of my colleagues and mentors over the years. Leeya Engel has been a most wonderful mentor and friend, ever ready to help me troubleshoot my protein patterning troubles. Erica Castillo, Liam Dow, Jens Moller, Sasha Denisin, Robin Wilson, and Alexandre Ribiero all provided instruction and inspiration along the way. And my protein patterning postdoc friend Rob Nwokonko for just being someone I could share all my random science excitement and ramblings with.

I am in the most sincere gratitude to the administrative staff who have literally enabled me to survive as a non-traditional student at Stanford. Schantae Wright always looked after my best interests to help make sure that I didn't fall between the cracks as it can happen to so many PhD students at big universities. Rachelle Riley and Jzesern Tan made sure experiments were able to proceed smoothly by making sure we had what we needed when we needed it–and that means more to a PhD student than most can realize. I sincerely thank all of the admins in the Departments of Mechanical Engineering and Molecular and Cellular Physiology for their contributions. Even if it was just a friendly smile. Again, that means a lot to someone like me.

Additionally, my science friendships in the Mechanical Engineering in addition to other departments such as MCP Department and ChemE have more broadly helped to further my studies as an interdisciplinary scientist and enhanced my experience at Stanford. Thank you Cayla Miller, Carlos Garzon-Coral, Claudia Vasquez, Gaby Baylon, Alberto Arvayo, Marc Levenston, John Janetzo,

Erin Sanders, Lucy O'Brien, Meritt Maduke, Rich Lewis, Bill Weis, and Alex Dunn.

I thank my committee members for always being the very best advocates that a PhD student could hope for. I thank Tom Clandinin for stepping up to chair my oral defense. This was a great honor for me since his infectious passion for neurobiology continues to inspire me. My reading committee members Ovi Chaudhuri and Tom Kenny have been nothing but supportive in my journey and I'm so grateful to have these voices behind me. Tom personally made sure that I would continue to be supported as a student in the Department of Mechanical Engineering through thick and thin. His name is on the textbook that first sparked my curiosity about how mechanical force could be transduced into an eletric potential. I'll never forget when he called me to say I had been accepted to the program and was being considered for a fellowship. It was a dream come true in the most genuine sense of the phrase. I thank Ovi for his continued mentorship as a now member of my Postdoctoral Advisory Committee. I still have the print out copy of "Life at Low Reynolds Number" that he suggested I read and that set my perspective for thinking about the world from the cell's perspective. And to my second PAC member, John Dabiri. You may not realize it, but your mentorship has been an incredibly valuable source of guidance along my postdoc job search. Thank you for being an accessible human being.

And of course, none of this experience would have happened without my wonderful, amazing, and most appreciated lab mates: Sujay Guha, Lucero Rogel, Emily Fryer, Tessa Logan, Hodan Farah, Eileen Mazochette, Tom Larsen, Gaspard Pardon, Jason Caesar, Anna Kim, Ohi Dibua, Cheavar Blair, Orlando Chirikian, Miguel Garcia, Frederic Loizeau, and Chandi Jaisinghani.

This entire journey was made possible thanks to the educators who truly lifted me up along my journey from community college to Stanford. These were the ones who saw my small bit of potential combined with my enthusiasm for knowledge and called to arms. Thank you Drs. Amelito Enriquez, Katie Wilkinson, Adam Leeper, Paul Mitiguy, Fred Barez, Nicole Okamoto, Raymond Yee, Jinny Rhee, John Lee, Thalia Anagnos, and Buff Furman.

Among these, my first research mentor of two years, Katie Wilkinson, has played such a special role in my research career. This is the source of my passion for sensory physiology. Her excitement is seriously contagious and more than that, she's always encouraged my curiosity. That's important. Rather than shutting me down or discouraging my unusual ways of thinking, she helped me explore

them deeper. It's remarkable how much she and Miriam have in common that way. It's been a very lucky circumstance to be have trained with both. They have trained my thinking of neurophysiology and sensory transduction. This is everything I love about science and as such it's the biggest gift anyone could give me. Thank you both so very very much.

But life is much more than just science and engineering. It has been an 11 year journey and I never would have lasted this long without the incredible community behind me. They may not have helped with the experiments, but they kept me alive. Life has not always been so kind. It has been painful and when I've felt the most isolated, they've reminded me that I'm surrounding by shining stars. I won't be able to name all of you here, but to all of my Bay Area Bike Friends, my Stanford Cycling teammates, Rainer Zaechelein and the Menlo Velo shop and CX crew, Bernardo Tapia the Wolfpack, Paule and Julie Bates and all of Team Roaring Mouse, my CX ladies...I love ya'll so much.

I thank my girlfriends for the love they share. Alison, where would I be without you?!? Seriously? You make me a better person. All of you. Julie Walker, Murphy Temple, Sarah Meyer, Anusha Allawala, Courtney Vella. Jen Wilson.

Thank you, mi familia, for being my number one fans. I know it's not easy because being on this path means I'm not around as much as you'd like. But know that I love you so much and you inspire me to never give up. Mom. Ray. Sade. Amber. Nire. Irene. Mariah. Lena. Story. Tia Julie. Tio Robert. Greg. Nancy. My Franco family. Near and far. Thanks for cheering me on.

Last, but far from least, I thank my partner and puppy daddy. How amazing is it that we met in our first year at Stanford? How amazing is it that we have been through as much as we have? I am so thankful to have you by my side. Thank you for your infinite support, patience, forgiveness, and love. The fact that we survived through shelter-in-place together, in that little room, at that kitchen table, for more than a year is some absolute type of incredible. Thanks for being my best friend.

Well, maybe not the last. What type of puppy mommy would I be if I didn't thank my little ride or die, Einstein Von Valentine? I don't know how I would have survived all these years without you dude.

Thank you to those who believed in me, even when I didn't believe in myself. Even if I didn't name you here. Thank you.

Funding

The following funding sources made this research possible:

1. NSF GRFP
2. VPGE EDGE
3. School of Engineering Harold and Marcia Wager Fellowship
4. NIH DSPAN F99

Contents

List of Tables

List of Figures

Chapter 1

Introduction

-

1.1 Motivation

As humans we rely on mechanosensation for survival. We take in mechanical cues through our ears, skin, muscle, tendons, joints, and even our internal organs. Mechanosensation defines how we interact with our environment and ourselves. We move away from loud noises or rough surfaces. The sensation of fullness tells us to stop eating [1]. Even without us knowing, our vestibular and proprioceptive sensory systems detect piconewton scale forces within hundreds of milliseconds to keep us walking upright [2]. The underlying commonality of all these sensory processes is the rapid conversion of force or displacement, pushes and pulls, into neural impulses that is broadly referred to as mechanosensation [3] (Figure 1.1). This is enabled by the hundreds of thousands of sensory neurons, such as dorsal root, spiral, and vestibular ganglion, that innervate the various organs of our bodies [4]. Despite the fact that mechanosensation plays such a vital role in our daily lives, there is still much that we do not understand about how it works and why it fails. At the most rudimentary level, we know that a rearrangement of integral membrane proteins at the cellular level allows ions to flow across the surface of specialized sensory cells in response to mechanical stimuli [5, 6]. However, we have yet to fully answer even the most basic questions such as which proteins are involved? and how is mechanical propagated from tissue to molecules?[1] Progress in

this field of study has recently been accelerated by interdisciplinary efforts to combine mechanical engineering principles with sensory neuroscience. The focus of this thesis is to continue this effort by building tools to improve the measurement resolution required for studying mechanosensation at the molecular and cellular levels.

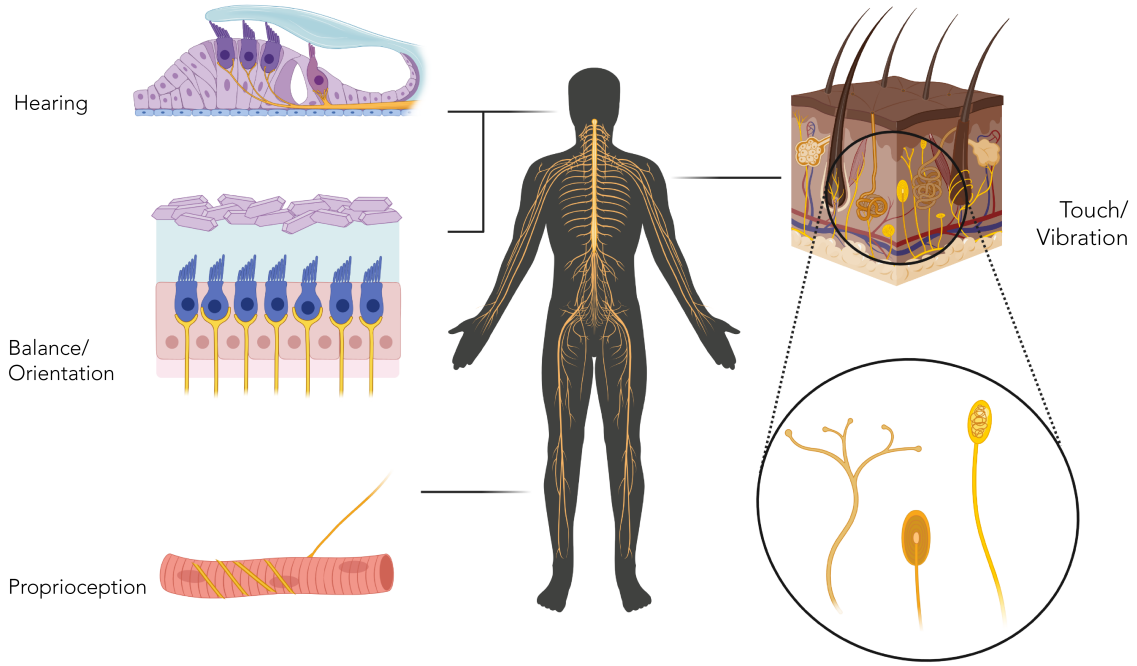
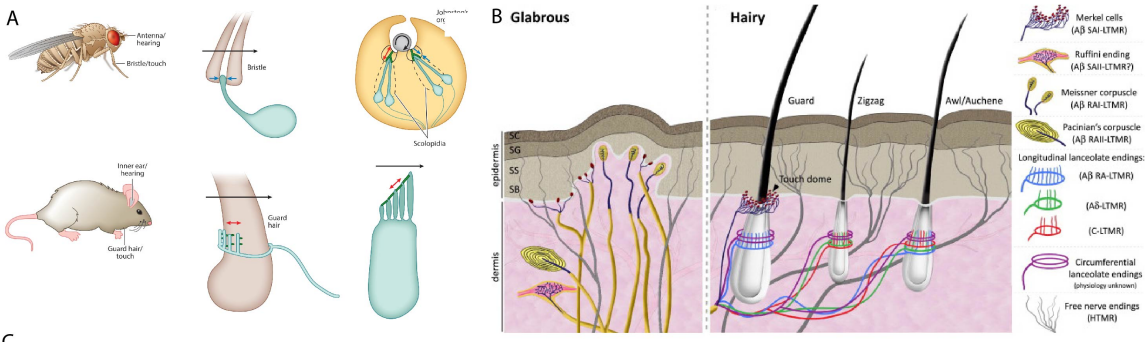


Figure 1.1: **Examples of mechanosensation in human**. Mechanosensation occurs via specialized sensory cells called mechanoreceptors in the auditory (hearing), vestibular (balance/orientation), and somatosensory (proprioception, touch/vibration) systems. Examples of mechanoreceptors in skin (zoom, left to right) include free sensory terminals or nerve endings, Pacinian corpuscles, and Meissner's corpuscles. Created with BioRender.com.

At a minimum, the prevalence of peripheral sensory neuropathies that interfere with mechanosensing warrants understanding the sensory process at the cellular and molecular scales. Peripheral neuropathies manifest as numbness, tingling, hypersensitivity, ataxias, weakness, vertigo, and hearing loss [7]. In neuropathies that target the somatosensory system, which is responsible for detecting gentle touch or feeling vibrations, seemingly harmless changes in sensation can develop into chronic pain that impairs mental health and quality of life [8, 9]. For

example, chemotherapy induced peripheral neuropathy is a common side effect of life saving cancer treatments that is so disruptive to daily life that it forces patients to stop chemotherapy [10]. Therapeutic targets that can successfully restore sensory function are yet to be identified [11], and this is likely a direct consequence of our limited understanding of mechanosensing at the molecular scale. It is practically difficult to fix a system without knowing how the system works. We can also consider the need to uncover the molecular basis for mechanosensation from a bio-inspired design perspective for use in engineered stress and strain sensors. Specialized sensory cells that detect mechanical stimuli, also referred to as mechanoreceptors achieve a remarkable functional specificity that has yet to be fully replicated in engineered systems. These biological strain gauges are characterized by a narrow dynamic range, finely tuned sensitivity and, in many cases, frequency selectivity that suggests they act as sensory band-pass filters [12]. The mechanisms that endow such properties to mechanoreceptors are not entirely clear, but there is surmounting evidence that they depend on the molecular composition and interactions, cellular architecture or morphology, and extracellular mechanics [13]. Examples of this functional and structural diversity are shown in Figure 1.2. These topics are the central motivations for the work presented in Chapters 3 and 4 of this thesis, and thus those chapters contain a more detailed discussion of the evidence supporting this hypothesis. In short, we have only recently begun to develop a more comprehensive view of the mechanosensation at the initial site of sensory transduction, as it occurs via specialized molecules and cells.

While mechanosensation is an important part of the human experience, it is not unique to humans. Creatures big and small, vertebrate and invertebrate alike, are able to respond to mechanical cues such as touch. The ubiquity of mechanosensation throughout the phyla further motivates studying the sensory process through the lens of bio-inspired design. Animals in nature show incredible sensory prowess that goes beyond what we experience as humans. For example, elephants use the Pacinian corpuscles, a type of mechanoreceptor, embedded in their foot pads to detect low frequency vibrations in the ground that help them locate their family members [14]. The non-parasitic roundworm *Caenorhabditis elegans* uses low threshold mechanoreceptors to detect sub-micronewton forces from threatening fungi that seek to trap the worm in a noose-like structure [15]. Most remarkably, many of the genes involved in producing the molecules necessary for touch



C

Physiological subtype	Associated fiber (conduction velocity[1])	Skin type	End organ/ending type	Location	Optimal Stimulus[4]	Response properties
SAI-LTMR	Aβ (16-96m/s)	Glabrous	Merkel cell	Basal Layer of epidermis	Indentation	
		Hairy	Merkel cell (touch dome)	Around Guard hair follicles		
SAII-LTMR	Aβ (20-100m/s)	Glabrous	Ruffini[2]	Dermis[3]	Stretch	
		Hairy	unclear	unclear		
RAI-LTMR	Aβ (26-91m/s)	Glabrous	Meissner corpuscle	Dermal papillae	Skin movement	
		Hairy	Longitudinal lanceolate ending	Guard/Awl-Auchene hair follicles	Hair follicle deflection	
RAII-LTMR	Aβ (30-90m/s)	Glabrous	Pacinian corpuscle	Deep dermis	Vibration	
Aδ-LTMR	Aδ (5-30m/s)	Hairy	Longitudinal lanceolate ending	Awl-Auchene/ Zigzag hair follicles	Hair follicle deflection	
C-LTMR	C (0.2-2m/s)	Hairy	Longitudinal lanceolate ending	Awl-Auchene/ Zigzag hair follicles	Hair follicle deflection	
HTMR	Aβ/Aδ/C (0.5-100m/s)	Glabrous Hairy	Free nerve ending	Epidermis/Dermis	Noxious mechanical	

Figure 1.2: **Functional and structural specificity of mechanoreceptors across phyla**. A. Mechanoreceptors in flies and mice used for detecting deflection of hair bristles (middle column) and sound (right column). Reprinted from [13] with permission from Elsevier. B. Mechanoreceptors in mammalian skin and (C) their structural and functional properties. B and C reprinted from [12] with permission from Elsevier.

sensation are conserved even when the size of the animal or its mechanoreceptors differ by multiple orders of magnitude [16, 17]. These similarities are also what allows us to focus our research efforts on studying mechanosensation in a model system that is best suited for the task, such as *C. elegans*. This nematode has an easily manipulated genome, fully mapped nervous system, is transparent, and also has a highly stereotyped behavioral response to gentle touch making it an ideal test bed for dissecting the molecular and cellular constituents of mechanosensation. In the remainder of this chapter, I focus on describing gentle touch sensation in the worm, and conclude by highlighting the gaps in our current knowledge and how this thesis contributes to filling those voids.

1.2 The cellular basis of gentle touch in *C. elegans*

1.2.1 Touch receptor neurons are low-threshold mechanoreceptors

C. elegans is a species of free-living, or non-parasitic, roundworms belonging to the Nematoda phylum (Figure 1.3.A). In nature the animals can often be found in moist soil or decomposing vegetation such as a pile of leaves or compost. It was first identified in the late 1800's as an ideal model for biological discovery, but was not widely embraced until the 1960s when its utility for studying genetics and development was established [18]. The animal is well suited for laboratory work, as it develops from an embryo to gravid adult in three days and is easily maintained on agar plates at standard temperature with innocuous *E. coli*. A full description of the animal's characteristics and usage is beyond the scope of this chapter and curious readers are directed to this excellent review for greater detail [18]. As previously mentioned, this animal has numerous properties that make it an ideal system for studying touch sensation. Chief among these is how the animal behaves when it is subjected to low-threshold mechanical stimuli, such as the application of force with an eyebrow hair or the "touch test" as it is called. Applying very gentle pressure to the animal will elicit an avoidance response that can easily be scored to identify deficits in the mechanosensory process following mutagenesis. This behavioral response can either be an increase in the animal's speed or a reversal in the animal's direction (Figure 1.8). The average forces applied by an experimenter during these tests are slightly greater than one micronewton, which is roughly 4x greater than the required force to activate the low-threshold mechanoreceptors (LTMRs) that detect gentle touch [19] (Figure 1.3.C). It was through these forward genetic screens that researchers identified a set of neurons that are necessary and sufficient to evoke the avoidance response, collectively known as the touch receptor neurons (TRNs, Figure 1.3).

The TRNs are oriented such that the longest extension of the cell, referred to as a neurite since it has properties of both axons and dendrites, runs along the anterior-posterior axis of the worm (Figure 1.3.B). The set of six TRNs can be further classified into pairs based on their anatomical positions: anterior lateral (ALM left and right), posterior lateral (PLM left and right), and anterior/posterior ventral (AVM and PVM). The animal moves forwards and backwards by dorsal/ventral contractions, similar to how a dolphin swims (Figure 1.3.A). Thus, when the animal is gliding along an

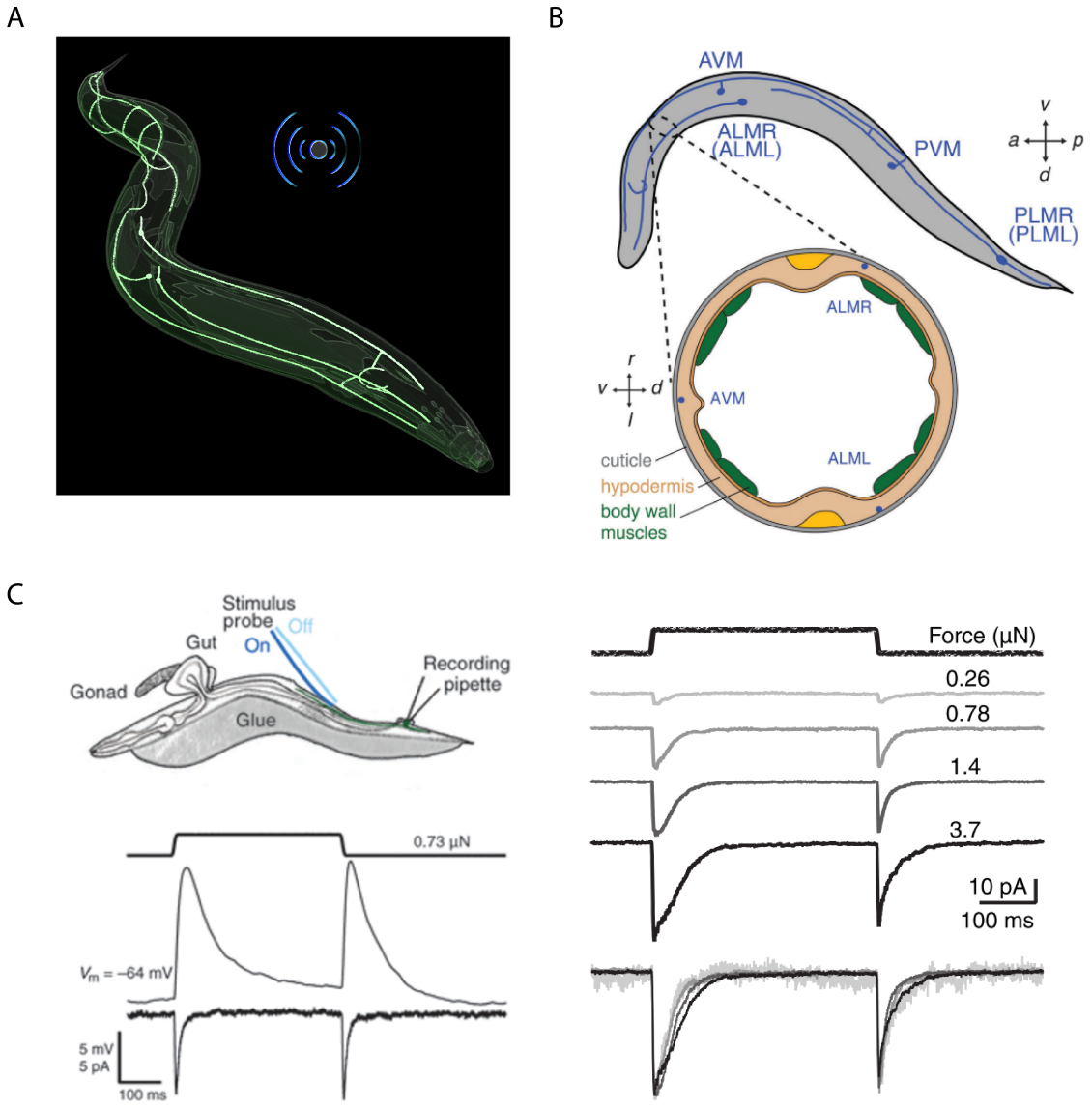


Figure 1.3: **Touch receptor neurons are low-threshold mechanoreceptors**. A. Perspective view schematic of *C. elegans* and the TRNs (brightest lines) during locomotion in response to vibration. Adapted from OpenWorm.org, licensed under the Creative Commons Attribution 3.0 Unported License. B. Defined anatomical orientations for the worm (ventral<->dorsal, anterior<->posterior) for the worm and cross-sectional view of the animal's body showing the location of the neurites just below the cuticle, engulfed by the hypodermal cells. Reprinted from [20] with permission from Oxford University Press. C. Whole cell patch clamp recordings from the TRN *in vivo* during mechanical stimulation and accompanying mechanoreceptor currents. Reprinted from [21] with permission from Springer Nature.

agar surface during a touch test, either the left or right side of the animal is exposed to the eyebrow hair. As a result, the ALM and PLM are often the focus of TRN research. Additionally, of the two ventral TRNs, only the AVM is required for touch sensation [16]. The PVM is categorized as a TRN partly owing to its nearly identical genetic and molecular similarities to the other neurons, but also for historical purposes. The ALMs are usually unipolar cells, although they can have a shorter, posteriorly oriented neurite giving them a PLM-like appearance. Though they are considered unipolar, the ALM does have small secondary branches that extend to the nerve ring near the pharynx of the animal. The AVM is the only pseudo-unipolar neuron of the set. It's primary branch extends circumferentially away from the cell body before forming secondary branches that run in both the anterior and posterior directions. All of the TRNs reside just below the cuticle of the animal and are engulfed by hypodermal cells. This arrangement shares many anatomical similarities to the free nerve endings that innervate human skin. The neurites themselves are extremely narrow in cross-sectional diameter. Though the exact number can vary along the length of the neurite, it is usually in the range of 200 nm [22, 23]. Eliminating the TRNs, either by genetic or laser ablation, abolishes the animal's response to gentle touch and establishes their necessity for the behavioral response. Similarly, optically activating the TRNs via ectopic expression of a channelrhodopsin evokes avoidance behavior, establishing that the TRNs are sufficient for mechanosensation of gentle touch [24].

All mechanoreceptors have functional specificity, as described earlier in this chapter, that allows them to respond to a narrow range of stimuli. This is how specific sensory inputs can be mapped to specific behavioral outputs. For example, when the animal is moving and its body undergoing dorsal-ventral contraction, it's critical that the AVM does not become activated due to the bending motion. If it did, the animal would not be able to distinguish between motion-derived inputs and those from a potential predator. *C. elegans* have three main types of mechanoreceptors that are all tied to different mechanosensory processes. SMD neurons within the animal act as proprioceptors that function as part of the sensory-motor circuit responsible for animal motion. The PVD and FLP neurons alert the animal to high-threshold stimuli which is also known as "harsh touch". Males of the species possess a fourth set of mechanosensory neurons that innervate their tails, which they use to locate the vulva of hermaphrodites for insemination [16, 25]. Historically the difference between harsh and gentle touch was derived from the apparatus used during the touch test. For a harsh touch

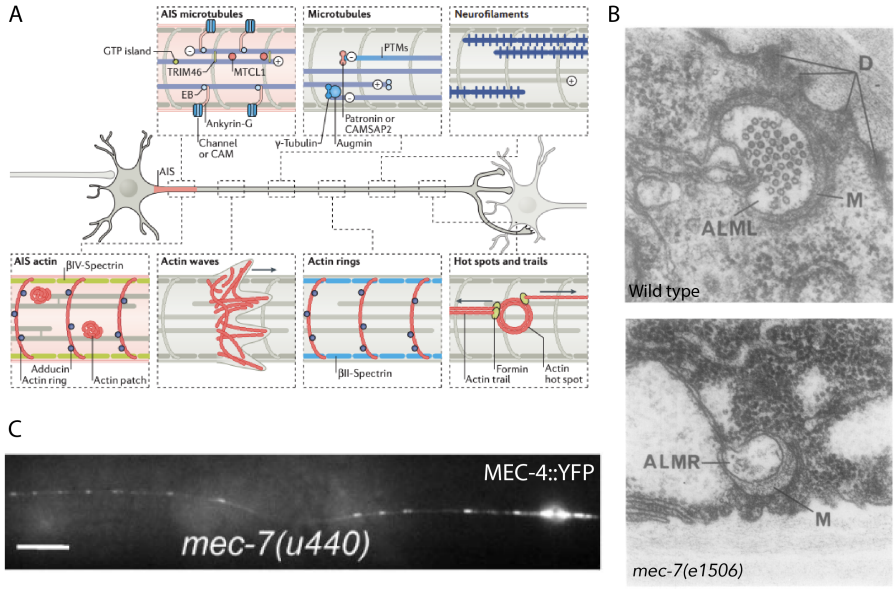
test, the same stroking motion is made with a stiff wire. Theoretically this should result in greater force transfer to the mechanoreceptors than the case of a soft eyebrow hair. In recent years as greater effort has been placed into quantifying and characterizing the mechanics of force transduction, these two types of touch can be thought of in terms of the amount of mechanical energy delivered to the neuron. The minimum amount of mechanical energy required to saturate the response of the neuron can then be used to classify the neurons as either low or high-threshold mechanoreceptors. This terminology is more closely aligned to that used when describing the afferents that innervate and detect touch in mammalian skin. Since TRNs require relatively low levels of magnitude of force for activation they are considered low-threshold mechanoreceptors

1.2.2 The TRN's cytoskeleton is necessary for sensory transduction

What structures within and near the TRNs are necessary for mechanosensation? Although a full discussion of these complex and fascinating structures is beyond the scope of this thesis, a short review will help to motivate the micropatterning work described in Chapter 3. The morphology of all animal cells as observed from outside the cell is a result of the underlying cytoskeleton that confers structural rigidity to the otherwise soft and flexible membrane. This network of rigid polymers can be thought of as the tent-poles that hold up the fabric of the cell. In the axons of neurons, or neurites of TRNs, the cytoskeleton is primarily comprised of microtubules that look like long poles running the length of the process (Figure 1.4). The nanoscale organization of the cytoskeleton in *C. elegans* TRNs share many similarities with mammalian axons as observed in these studies [23, 26]. An in depth review of axonal cytoskeletons can be found at this reference [27] Microtubules are also considered the "highways" of neurons, as they act like railroad tracks from motor proteins that traffic essential cargo from the soma to the distal regions of the process. The TRNs were originally named after the microtubules since these hollow, tube-like structures occupy the bulk of the intracellular space (Figure 1.4.B). The main focus of Chapter 3 is the development of protein micropatterning strategies for controlling the morphology of the TRNs to reduce heterogeneity in cytoskeletal arrangement that may interfere with our analysis of protein localization along the neurite. The results presented in Chapter 4, support the hypothesis that our protein of interest, MEC-4 (described in further detail below) is also trafficked along the neurite. This might explain why mutations to one

of the genes necessary for the formation of the microtubules, *mec-7*, can impair the distribution of MEC-4 puncta along the neurite (Figure 1.4.C). Animals with null mutations to the *mec-7* gene (*e1506)* show a complete loss of microtubules at the adult stage (Figure 1.4.B, lower panel). This further confirms the need to consider the role of the cytoskeleton in mechanosensation at the cellular level.

Many have considered what role the cytoskeleton may play in the activation of the ion channels necessary for mechanosensation (described below). The force-from-filament model encompasses the idea that the mechanoreceptor's cytoskeleton could serve as a physical linkage to the channel, acting as a conduit for force transduction [13, 29, 30]. This hypothetical gating mechanism is often a source of debate amongst biophysicists, as some of the identified mechanosensitive channels can be gated by membrane tension alone following the purported removal of the cytoskeleton [31, 32]. Two of the earliest identified gene products necessary for touch sensation are MEC-7 and MEC-12 which encode the *C. elegans* orthologs of beta- and alpha-tubulin subunits, respectively. These two tubulin proteins are structural constituents that make the 15-protofilament microtubules that are unique to the TRNs [33] (Figure 1.4.B). Mutating the genes that encode these proteins results in significant mechanosensory abnormalities and impaired mechanoreceptor currents (MRCs) [21, 22], which is consistent with the model that microtubules may act as intracellular tethers in channel gating. However, more recent direct and quantitative analysis of data from serial section electron microscopy thwarted this notion, as the distance between the ends of the microtubules and the ion channels is practically too large [23]. This same study found evidence that these microtubules are cross-linked at their distal tips to the plasma membrane, which leaves the possibility that the microtubules could be indirectly involved in force propagation along the neurite. Since mutations to the *mec-7* gene disrupts MEC-4 puncta, resulting in much larger fluorescent aggregates spaced further apart than in the wild type background [28] (Figure 1.4.C), suggesting the touch insensitivities could be the result of impaired channel distribution. At this point it isn't possible to draw conclusions regarding the precise role of microtubules in touch sensitivity. It will be interesting to see via finite element modeling that is linked to channel activation [34] how forces applied to one end of the microtubule bundles might propagate along the plasma membrane. Recent work analyzing other cytoskeletal components, such as beta-spectrin (UNC-70) and tau (PTL-1), have shown similar changes in touch



E

Genotype	Touch response (*mean±SD*)	*# of worms*
control (*uIs31[mec-17p::GFP]*)	9.35 ± 1.15	75
unc-70(e524);uIs31	6.33 ± 2.12*	203
ptl-1(ok621);uIs31	8 ± 1.4*	50
ptl-1(ok621);unc-70(e524);uIs31	5.3 ± 2.65***	75
ptl-1(ok621);unc-70(e524);mec-7(e1343);uIs31	1.5 ± 1.2***	25
mec-7(wy116);unc-70(e524);uIs31	4.3 ± 0.77***	50
mec-7(wy116);uIs31	8.6 ± 0.6^{ns}	142
mec-7(wy116);ptl-1(ok621);uIs31	9.5 ± 0.3^{ns}	154
mec-7(e1343);uIs31	1.2 ± 1.3***	137
ptl-1(pg73[ptl-1::mNeonGreen])	9.6 ± 0.77^{ns}	50

Figure 1.4: **Contributions of the axonal cytoskeleton to mechanosensitivity.** A. Organization of the axonal cytoskeleton. Reprinted from [27] with permission from Springer Nature. B. Electron micrograph showing the cross-sectional view of ALM in wild type and null mutant (*mec-7(e1506)*) backgrounds. Reprinted from [22] with permission from Academic Press. Adult animals without *mec-7* lack microtubules. C. Distribution of MEC-4::YFP puncta is disrupted in *mec-7(u440) animals*. Reprinted from [28] with permission from Elsevier. D. Beta-spectrin (*UNC-70*) organization is conserved in TRNs. E. Mutations to cytoskeleton encoding genes impairs touch sensitivity. D and E reprinted from [26] with permission from eLife under the Creative Commons Attribution license.

sensitivity that are often accompanied by changes in cellular mechanics [26, 35]. Only future studies can tell if the observed changes in mechanosensation are due to trafficking, mechanical, channel anchoring defects or some other mechanism not yet known.

1.2.3 *In vitro* approaches for studying TRNs

Though the first account of *C. elegans* being used as a specimen for biological studies was published in the 1ate 1800's [36], the first documentation work with its isolated cells (*in vitro*) would not be published until nearly a century later. Chief among these challenges is that there is no immortalized cell line available, and that primary cells must be prepared from either larvae or embryos. As acute dissociation of larvae and adults yield neurons without intact processes, this approach is not as amenable to mechanosensitivity assays and will not be discussed here. Acquiring isolated cells from dissociated *C. elegans* embryos is a straightforward process that can be completed within three hours using readily available lab equipment and reagents. The resulting cultures can be stored at standard temperature and humidity, without active carbon dioxide regulation. The most attractive aspect of this system is the ease with which transgenes can be expressed in isolated cells, as they simply express the parent strain's genome. Animals stably expressing transgenes are ready for use in primary culture within two weeks, resulting in isolated cells that can express the same fluorescent reporters or genetic mutations as the parent generation. These factors combined, this cell culture system greatly reduces the overhead associated with traditional methods while providing greater access to sophisticated genetic, mechanical, and live cell imaging experiments. Though there are challenges to this experimental approach, the substantial advantages over traditional cell culture techniques warrant its consideration. Leveraging the genetic tractability of the worm in single cell mechanosensitivity assays would open doors for new avenues of research to gain mechanistic understanding of the biophysics of channel gating. This thesis has focused on developing technical approaches for overcoming the challenges that have limited the feasibility of conducting these assays. To honor the work of those who laid the foundation for these studies, I will provide a brief account of the major contributors to our current understanding, and what their studies have taught us about the cells in culture.

Laird Bloom's PhD thesis, published in 1993, was the first available protocol for generating

cultures of isolated cells from *C. elegans* dissociations. Bloom recognized that one significant challenge with understanding how extracellular cues influence neurite outgrowth was a lack of tools for controlling the neural environment. At the time of his writing, there existed a single account of cultured embryonic cells, however there was no information provided with regards to experiment methods such as how the cells were obtained or culture conditions [37]. It would be another twelve years before this same group published a brief description of their culture methods and further described the excretory cells and neurons present in the cultures [38]. In his thesis, Bloom cites personal communication with Lois Edgar, who had developed a technique for removing the protective eggshell via chemical and mechanical methods. Edgar observed that these embryos could continue developing without the eggshell, reaching the 500 cell stage and occasionally displaying cells with neuronal-like processes. These methods had been optimized for culturing *C. elegans* blastomeres, and would shortly thereafter be published as such [39], but Bloom found that they worked equally well for early to mid-stage embryos that could be dissociated into individual cells [40]. Edgar's techniques were further refined for the culture of cells from dissociated larvae, however, the dissociation of whole animals inevitably disrupted neuronal processes and is thus less amenable to *in vitro* mechanosensation assays. Although both Edgar and Bloom had laid the groundwork for the embryonic dissociation technique it would be another seven years before a detailed protocol was available [41].

At the start of the 21st century two separate studies not only published thorough details of methods for dissociating the embryos of *C. elegans* and culturing the resulting single cells, but also demonstrated the enormous capabilities of this system. Today there are at least ten published protocols for working with these primary cultures, having over 1000 citations between them. Nearly 500 of these citations belong to the first two papers from the Chalfie and Strange groups, respectively [41, 42]. Surprisingly only a small subset (fewer than 50), as analyzed via a primary references literature review of citations containing the phrase "*in vitro elegans*", are published studies in *C. elegans*. That is, the work demonstrated by these papers, using this system, had significant impact on work beyond the *C. elegans* field. The Chalfie group published an innovative approach to genetically profiling purported neuronal subtypes by use of fluorescence activated cell sorting (FACS) of isolated *C. elegans* embryonic cells expressing GFP under a mechanoreceptor specific promoter [42]. This

pioneering work inspired researchers to think more deeply about variations in genetic expression between cellular phenotypes. Subsequent studies used a similar approach to create profiles for thermosensory, GABAergic, and motor neurons in these cultures as well mapping the transcriptome of muscle cells [43–46]. The Strange group demonstrated that these cultures could be used not only with FACS but also with electrophysiology experiments. They characterized the genetic profile of the total population and noted the presence of multiple neuron types and muscle cells. Additionally the genetic expression of these cultures was susceptible to interference methods such as the use of double stranded RNA [41]. Since that second generation of protocols became available, at least 50 papers have used this approach to conduct a number of important biological queries such as neurite outgrowth, membrane and cellular mechanics, glycobiology, neurodegeneration, electrophysiology of single cells, and protein trafficking [24, 35, 47–51].

1.3 The molecular basis of touch sensation in *C. elegans*

The ease with which the mechanosensory function of *C. elegans* can be scored gives researchers a powerful tool for identifying the genes necessary for touch sensation. In forward genetic screens, the DNA of animals is damaged by chemicals or radiation and the progeny of these animals are evaluated for mechanosensory abnormalities. By sequencing the DNA of animals who show insensitivity to gentle touch, the mutated genes responsible for the deficit are identified and subsequently named with the *mec* prefix, which is shorthand for "*mec*hanosensory abnormality" [52]. The set of *mec* genes are numbered in the chronological order in which they were discovered. Today, there are 17 genes known to be necessary for touch sensation, but only a subset of these encode molecules that are thought to be directly involved in the mechanotransduction process. In many cases the precise function of those gene products has yet to be determined. By analyzing the structure of gene products and drawing comparisons with other known molecules we can make predictions about their roles in the mechanosensory process. In this section I specifically focus on the molecules encoded by those genes that are thought to play a direct role in the rapid conversion of mechanical stimuli into neural impulses. Interested readers may find greater details of all *mec* genes (*mec-1* through *-19*, but not *-11*, *-13*, or *-16*) at the website WormBase.org. For organizational purposes I divide this

discussion based on the subcellular localization of the gene products, as this helps us construct a biophysical model of the molecular basis of mechanotransduction. This conceptualization is based on the established theories of force transduction via specialized ion channels. The assertion of this thesis is that the detection of gentle touch in *C. elegans* relies on the ECM to transmit mechanical energy to mechanoelectrical ion channels in the TRNs. This mode of activation is referred to as force-from-filaments [13] which may act in parallel with force-from-lipids. In that way, this biophysical model of mechanosensation provides design criteria for the work described in subsequent chapters.

1.3.1 Mechanoelectrical transduction (MeT) ion channels

Ion channels are protein complexes that form gated pores in the plasma membrane of all cells. Ion flow across the cell membrane is dependent upon the selectivity of the channel, the pore size, the driving potential, and the channel's activation state. For example, under certain conditions a sodium channel may be blocked by a drug and thus even if the channel is in an active state, no ions may flow across the membrane. Ion channels open and close in microseconds or faster and stay open for milliseconds, resulting in rapid cellular signaling events. A common feature of all mechanoreceptor cells is that they rely on the activation and gating of specialized ion channels. In the case of the TRNs, the relevant mechanoelectrical transduction channel (MeT) is a heterotrimer of conserved degenerin epithelial sodium channel (DEG/ENaC) proteins [53, 54] (Figure 1.5.B). The exact stoichiometry of the MeT channel remains unclear, but two of the channel subunits are the *mec* gene products MEC-4 and MEC-10. Both of these proteins are members of the DEG/ENaC family and will form homomeric channels when heterologously expressed in *Xenopus oocytes*. Regardless of the exact configuration, MEC-4 is is necessary for touch sensation and MEC-10 is dispensable. Though the *mec-4* gene was cloned in 1991, it was not until the advent of a method for recording electrical signals from the TRNs while also applying a mechanical stimulus to the worm that its function as a pore forming subunit was established [21].

The first evidence that MEC-4 might be part of a channel complex came from analysis of rare, gain-of-function alleles of the *mec-4* gene that caused degeneration of the TRNs. Animals expressing these mutated isoforms, referred to as *mec-4(d)*, have TRNs that are vacuolated and swollen

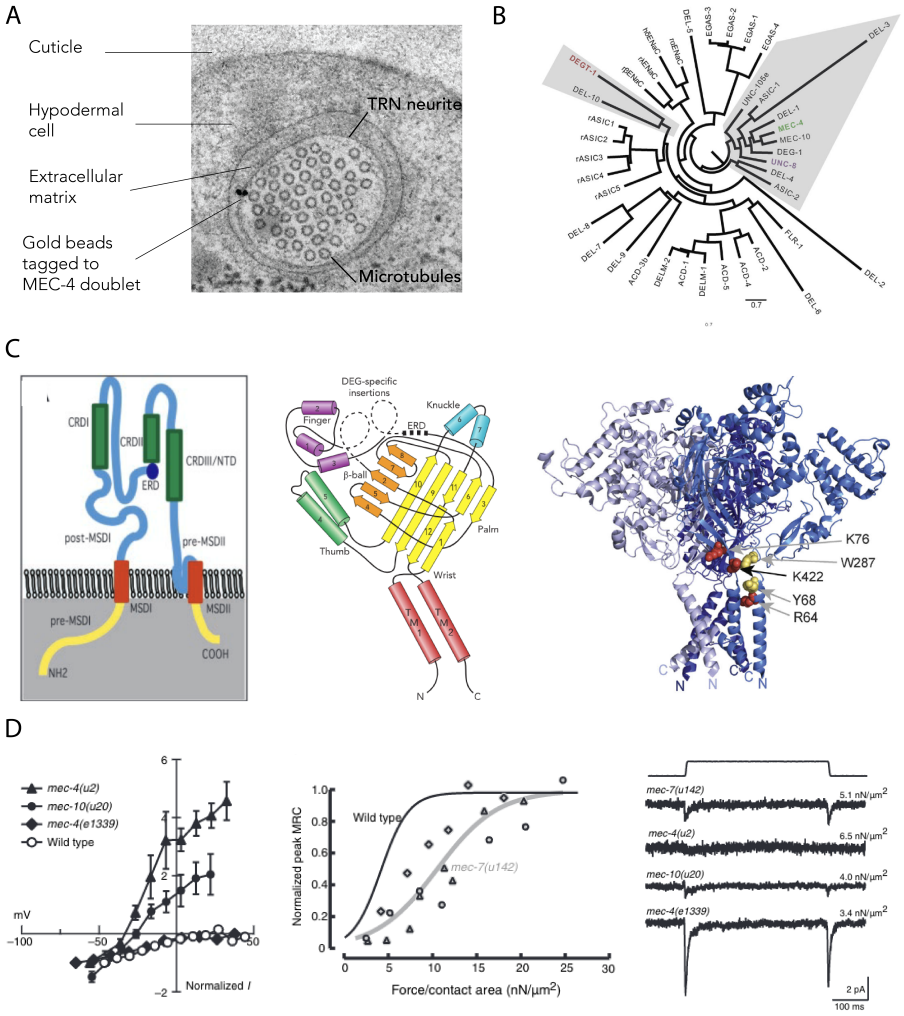


Figure 1.5: **Properties of the MEC-4 MeT ion channel**. A. Immunogold labeled electron micrograph stained with gold beads conjugated to MEC-4 antibodies showing the localization of MEC-4 to the plasma membrane of the TRN as a doublet. Reprinted from [23] with permission from Society for Neuroscience. B. Phylogenetic analysis of the DEG/ENaC/ASIC family to which MEC-4 belongs. Reprinted from [54] with permission from Rockefeller University Press. C. Structural analysis of MEC-4 (left and middle) and closest solved structure (right). Left reprinted from [55] with permission from Society for Neuroscience. Middle reprinted from [53] with permission from American Physiological Society. Right reprinted from [54] with permission from Rockefeller University Press. D. Electrical properties of the MeT channel in wild type and various genetic backgrounds. Reprinted from [21] with permission from Springer Nature.

from an osmotic imbalance [56]. Substantial efforts to predict the function of the protein based on a functional studies of its various domains over a ten year period provided further evidence that MEC-4 formed a MeT channel. These studies resulted in a predicted protein structure characterized by an unusually large extracellular domain, flanked by a transmembrane and intracellular domain on each side suggestive of an ion channel subunit [55]. Around the same time, recordings of *Xenopus oocytes* heterologously expressing the *mec-4(d)* gene showed MEC-4(d) formed an ion channel at the plasma membrane. This indicated that at a minimum, the degenerative form of the protein can form a homomeric channel on its own. Parallel efforts to use a genetically-encoded calcium indicator expressed under the *mec-4* gene to record calcium transients from the TRNs *in vivo* evoked by pushing a stiff probe in the animal's body demonstrated that these neurons are bona fide mechanosensory neurons. This study also established that *mec-4* null animals failed to generate calcium transients in response to mechanical stimuli [57]. While this further supported the hypothesis that MEC-4 is part of a channel complex, these experiments could not directly test MEC-4's involvement in forming the channel directly activated by mechanical stimuli.

The final piece of evidence that established MEC-4 as a pore-forming subunit of the MeT ion channel was obtained via *in vivo* electrophysiological recordings in response to mechanical stimulation. No group had previously recorded from *C. elegans* TRNs, as the dissection and patch clamping techniques were not developed until the late 1990s [58]. When paired with an open-loop system that applied force to dissected animals via a flexible probe, this experimental setup allowed for the first ever direct recordings of mechanoreceptor currents (MRCs) generated by the endogenously expressed MeT ion channels *in vivo*. Applying a step-like stimulus of less than a microNewton of force via the stimulus probe to the TRN results in a membrane depolarization and accompanying MRCs. MRCs increase in size with the applied force. In this paper it was also clearly demonstrated that sodium is the primary charge carrier responsible for these MRCs and that these currents could be blocked with the drug amiloride. Consistent with previous reports, *mec-4* null animals showed a complete loss of force-evoked MRCs. Most importantly, the group showed that MEC-4 and MEC-10 are both pore-forming subunits of the MeT channel demonstrating that MRCs reversed direction at positive voltages in wild0-type and negative voltages in mutants encoding point mutations in MEC-4 and MEC-10 (Figure 1.5). This established that MEC-4 and MEC-10 are both constituents

of the MeT channel that produces rapid membrane depolarizations in response to gentle touch, and that this MeT channel cannot be activated simply by altering the membrane potential.

MEC-4 has a distinctive sub-cellular expression pattern *in vivo* that has been demonstrated by many groups using both antibodies against the protein itself and multiple fusion-protein constructs [23, 28, 59–61]. Briefly, MEC-4 forms distinctive punctate structures along the neurite with little variation (Figure 1.6). The methods for quantifying the distribution have evolved over the years, but a detailed analysis points to reasonable explanations for small differences in puncta spacing [61] (Figure 1.6.C). This patterning was first reported using a yellow fluorescent protein fused directly to MEC-4 and the puncta were observed in the ALM, PVM, and PLMs [28]. Subsequent studies using antibodies raised against MEC-4 allowed for immunolabeling of the endogenous protein *in vivo* using fluorescent probes as well as gold beads in serial-section electron microscopy experiments. These studies provided direct evidence that MEC-4 localizes to the plasma membrane of the TRN, that the predicted extracellular domain is indeed ECM facing, and that the MeT channels may vary in stoichiometry. In the serial-section immunogold labeling experiments, gold beads with antibodies against MEC-4 were sometimes found in doublets and sometimes as single beads along the length of the neurite [23] (Figure 1.5). The findings from this study also support the hypothesis that each of the observed fluorescent puncta comes from a single MeT channel. Though these puncta are necessary for gentle touch sensation, they are not alone sufficient to evoke the avoidance response, as will be discussed in the following section. The aforementioned studies point to complex genetic interactions between *mec-4* and several other *mec* genes that are required for both formation of the puncta and proper TRN activation.

1.3.2 Contributions of the membrane to mechanosensation

The biophysical mechanisms that enable MEC-4 channel activation have yet to be fully uncovered and contributions from the lipid membrane cannot be ignored. Many of the other known mechanically gated ion channels can be activated via forces delivered to the channel from the surrounding membrane [62] in what is called “force-from-lipid” mode [63] (Figure 1.7). There is evidence that proper MEC-4 channel activation requires contributions from the surrounding lipids [24, 64], although the details of this relationship are still emerging. Here I will briefly discuss two

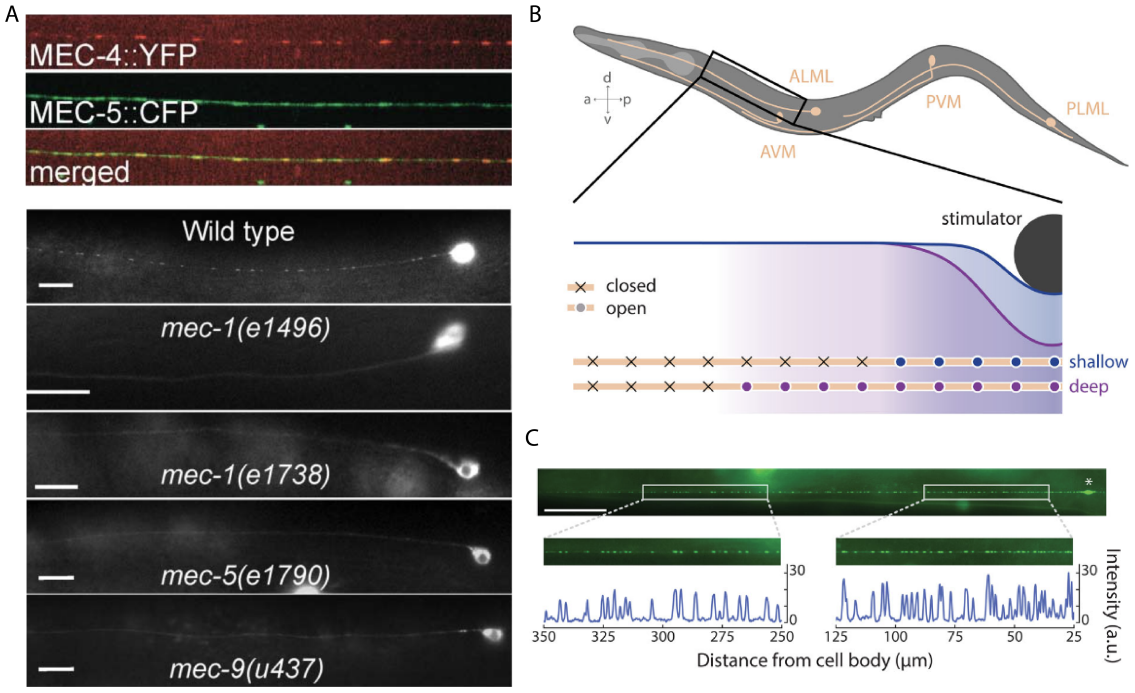


Figure 1.6: **MEC-4 localization and progressive recruitment during stimulation**. A. MEC-4::Yellow fluorescent protein (MEC-4::YFP) fusion construct is expressed as distinct puncta along the TRN neurite in an ECM dependent manner. Reprinted from [28] with permission from Elsevier. B. Biomechanical model of progressive recruitment of the MeT channel during mechanical stimulation. C. mNG::MEC-4 localization along the TRN and quantification strategy showing increased puncta spacing at distal regions of the neurite. B and C reprinted from [61] with permission from Rockefeller University Press.

points of interest with regards to force transduction to the MEC-4 channel via the plasma membrane. First, I will describe the evidence that altering membrane composition alters channel activity. Then, I will discuss a potential mechanism for force transduction from the lipids to the channel via the *mec-2* gene product that is necessary for mechanosensation. A more detailed description of the biophysical principles enabling channel gating can be found at ref [65].

The plasma membrane of cells, in which ion channels reside, are diverse structures rich with a heterogeneity of components and organization. There is growing evidence that these membranes have specialized regions, and that these regions along the surface of the membrane contribute to the function of membrane bound structures such as ion channels [66]. The classic textbook representation of the plasma membrane is the lipid bilayer primarily composed of glycerophospholipids drawn as a hydrophilic spherical head and two hydrophobic tails. The tails meet each other at the inner leaflets of the membrane, while the heads are exposed either to the intracellular or extracellular milieu [67]. Within these membranes, even in the case of the TRN, exist poly-unsaturated fatty acids (PUFA) that are known to regulate MeT channel gating via force-from-lipids [68]. In the TRNs, two PUFAs in particular, arachidonic and eicosapentaenoic acids contribute to touch sensitivity. These PUFAs are synthesized via a complex pathway that depends on the genes *fat-1* and *fat-4*. Animals with mutations to these genes show a disrupted response to repeated touch. Interestingly, this study also tested the membrane mechanics of *fat-1;fat-4* mutants by using atomic force microscopy (AFM) to pull membrane tethers from the surfaces of cultured TRNs. They found that membranes from cells with the PUFA mutant background were more resistant to membrane tether extrusion at increasing velocities. This fits within the mechanical model of PUFAs being necessary for maintaining the membrane thickness and bending stiffness required for wild type touch sensitivity [24]. This elucidated a potential contribution of the membrane to the mechanosensing process.

Lipids are molecules that, from a biophysical standpoint, would not serve as an efficient mechanism for force transduction due to their fluid-like mobility within the membrane. If the MEC-4 channel senses force-from-lipids, it would likely require some type of linker. The most obvious candidate is the stomatin-like protein encoded by the *mec-2* gene. The *mec-2* gene was first identified as being necessary for touch sensation in initial screens of mutagenized animals [56]. Through heterologous co-expression of MEC-2, MEC-4 and MEC-10 in *Xenopus oocytes*, researchers showed

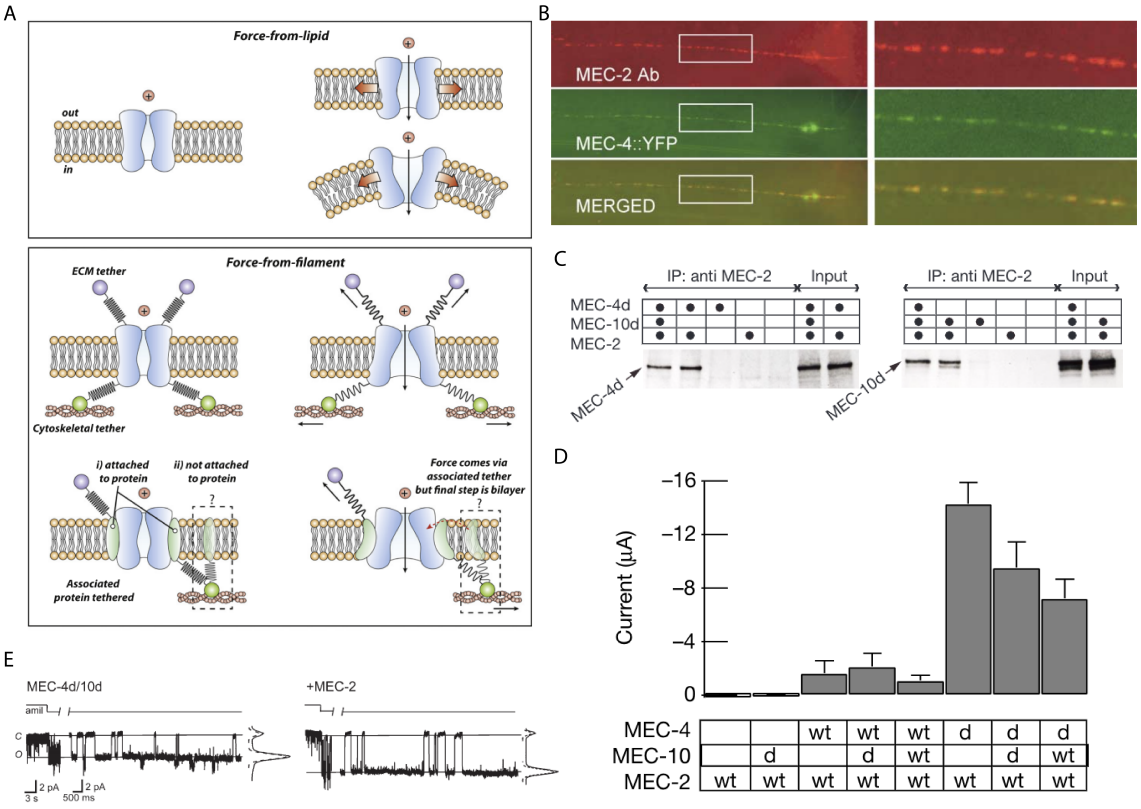


Figure 1.7: **Models of MeT gating mechanisms and evidence of contributions from the membrane**. A. Proposed biophysical models of MeT channel gating by force-from-lipid or -tethers. Reprinted from [62] with permission from Elsevier. B. Co-localization of the cholesterol binding protein MEC-2 with MEC-4::YFP along the neurite. Reprinted from [69] with permission from Cell Press. C. Co-immunoprecipitation of MEC-2 with MEC-4(d)/10(d). D. Co-expression of MEC-2 with MEC-4/10 (wild type and degenerin, d) increases currents in *Xenopus oocytes*. C and D reprinted from [70] with permission from Nature Research. E. The addition of MEC-2 increases single channel conductance of MEC-4/10(d) channel in *Xenopus oocytes*. Reprinted from [71] with permission from Rockefeller University Press.

that MEC-2 dramatically increased the currents without altering surface expression of MEC-4(d) or MEC-10(d) (Figure 1.7.D). That is, the total current was increased but not because of there being more ion channels at the surface. In that same publication, researchers showed MEC-2 physically interacts with MEC-4(d) and MEC-10(d) in co-immunoprecipitation experiments [70] (Figure 1.7.C), supporting the hypothesis that MEC-2 functions an accessory subunit. With the development of an

antibody against MEC-2, later experiments were able to show that MEC-2 colocalized with MEC-4 puncta along the TRN in a robust manner (Figure 1.7.A). These experiments also showed that the formation of MEC-4 puncta does not depend on MEC-2, as null animals have normal puncta [69] but animals expressing the same *u37* allele of *mec-2* lack MRCs as shown in [21]. This demonstrates that the presence of MEC-4 puncta alone is not sufficient for touch sensation and that these puncta require MEC-2 for channel activation. Later repeated studies of *Xenopus oocytes* expressing *mec-2* and *mec-4* showed that MEC-2 specifically increased the single channel conductance of MEC-4(d)/10(d) [71] (Figure 1.7.E). These results combined with further analysis of MEC-2 and its cholesterol binding activity [72] all point to a biophysical model of MEC-2 acting as a physical linker delivering force from the membrane to the MEC-4 channel.

1.3.3 The ECM is necessary for MEC-4 localization

The distinctive localization of MEC-4 to the plasma membrane in a punctate fashion has been shown to be necessary for touch sensation in every tested case. To date, no genetic background has been identified that results in an animal lacking MEC-4 puncta but showing a response to gentle touch. The first indication that the ECM plays an essential role in mechanosensation came from early forward genetic screens and follow up studies to identify gene product structure, function, and localization. The tools developed in this thesis were largely inspired by a desire to elucidate which aspects of channel localization were cell autonomous and which relied on cues from the ECM. Though there are only three known ECM proteins known to be required for channel distribution (Figure 1.6.A), Chapter 4 of this thesis identifies additional ECM proteins that may contribute to activation of MEC-4 channels. It is important to note that since the proteins described in this section are necessary for channel localization, it has not been feasible to test if they also serve as mechanical linkages transducing mechanical energy via force-from-tethers. As one might expect from first principles, mechanical stimuli will propagate to the TRN regardless of how it is attached to the ECM [73], but the question of whether this strain can result in channel activation without direct tethers remains unanswered.

Experiments using MEC-4 fluorescent protein fusions in various genetic backgrounds have all shown that MEC-4 puncta requires the genes *mec-1*, *mec-5*, and *mec-9* [28]. Of these, *mec-5* has

been most extensively characterized and encodes for a type of collagen (MEC-5) that is unique to *C. elegans* in terms of the number of glycine-X-Y repeats and other amino acids. Initial lac-Z reporter experiments showed clear expression of the *mec-5* promoter in hypodermal cells [74], but more recent single cell RNA sequencing data suggests the gene may actually be expressed in body wall muscle cells [75]. Regardless, these sources and others agree that *mec-5* is not likely expressed by the touch receptor neurons [47], despite even the most direct evidence that it preferentially localizes to the ECM surrounding the TRN neurite [23]. Conversely, *mec-1* and *mec-9* are almost exclusively expressed by the TRNs, with at least ten fold higher transcript copy numbers than the handful of other neuron types that might also be expressing these genes [75]. The MEC-1 and MEC-9 proteins harbor multiple Kunitz domains, suggesting that they may act as protease inhibitors to sculpt the ECM. The proteins also have a number of epidermal growth factor (EGF) like repeats that may allow them to bind to other ECM proteins. Initial studies suggested that MEC-1 and MEC-5 strongly colocalize with each other, and that these two may colocalize with MEC-4 and may be part of the mechanosensory complex [28]. However, later studies using serial-section immunogold labeling of MEC-4 and fluorescent probes with antibodies against the endogenous protein established that MEC-5 does not necessarily co-localize with MEC-4 or MEC-2 [23]. To date there have been no successful efforts to visualize the subcellular position of MEC-9. In any case, these studies all confirmed that genetic interactions between *mec-1*, *mec-5*, and *mec-9* are required for MEC-4 localization.

1.4 Tools for assaying touch

The original touch test described above was an essential tool for identifying mec genes and the functions of their products, but it lacked the resolution needed to tease apart more nuanced aspects of touch sensation. Measuring the response of the TRNs to controlled forces and displacements, with ability to apply scaled magnitudes, also allowed researchers to build a model of force propagation explaining how, at the macroscale, force propagates to the TRN. These studies have all focused on assaying mechanosensation *in vivo*, and the work of this thesis has centered on expanding this to

analyzing cell autonomous aspects of touch sensation. In Chapter 2, I describe preliminary experiments to assay touch sensitivity in cultured cells using combined atomic force microscopy (AFM) and fluorescence imaging of calcium transients. At the time of this writing only a handful of groups have assessed the mechanosensitivity of cultured TRNs [41, 76], but there has been little consensus with regards to the molecular basis of the observed TRN activation [77]. The methods used to perform these assays are common electrophysiological techniques that use suction to deliver membrane stretch and will not be covered in this section. The discussion here focuses on a handful of novel methods that directly informed our current model of force propagation to the TRNs in the most recent years. A more exhaustive discussion of the earlier methods available for assaying gentle touch in animals may be found at this excellent reference [25, 78].

Potentially the biggest breakthrough in the pursuit of modeling the mechanics of touch sensation came with the advent of a force-clamp system that allowed for the application of controlled forces to the worm's body [79] (Figure 1.8.A). Previous experimental approaches were limited to open-loop control systems where an applied force could be commanded, but there was no way to measure the actual force delivered and adjust for stress relaxation [21]. This new approach leveraged a self-sensing piezoresistive cantilever to measure the deflection of the cantilever throughout the course of force application. By routing this data back through a field programmable gate array (FPGA), the system could rapidly calculate the error between the commanded and actual forces. This type of measurement system is critical for measuring viscoelastic materials that exhibit creep in response to applied stresses. This also allowed researchers to measure indentation depths associated with the commanded force. Through these experiments the group discovered that body mechanics dominated the amount of stimulus delivered to the TRN. Animals with mutations resulting in softer bodies, such as *dpy-5(e61)* mutants, required less force to elicit the avoidance response. Conversely, those with stiffer bodies as a result of mutations to the *lon-2* gene required a greater amount of force. These experiments showed for the first time that indentation depth is the determining factor for TRN activation [20]. A similar system was further demonstrated in experiments measuring the response of the TRNs to vibratory stimuli that showed the TRN acts as a high pass filter, being most easily excited by higher frequency stimuli [80]. Combined, this points to a biomechanical model wherein

the proximal mechanical stimulus for TRN activation is neurite stretch and the extent of this deformation depends on stimulus speed.

These findings inspired new directions for technological advancement that would allow researchers to (1) visualize longitudinal strain within the TRN, and (2) apply mechanical stimuli in displacement-clamp mode at higher speeds. Two approaches were developed in parallel to meet these technical demands, each centered on a different method for visualizing TRN activation. The first aimed to measure TRN activation using calcium transients and so a microfluidic chip was used to immobilize animals while providing a clear view of the TRN with fluorescence imaging and applying stimuli via a pneumatic actuator (Figure 1.8.B). Interestingly, this work yielded an unexpected result. Calcium transients in AVM and PVM neurons were significantly greater than those of the ALM in response to a 275 kPa buzz stimulus with 75 kPa oscillations at 10 Hz [81]. For comparison, 10 Hz was the magnitude of the slowest frequency at which a detectable MRC could be recorded in force-clamp experiments [80]. This same device allowed for the steady-state tracking of mitochondria within the TRN of a living worm during a 300 kPa pressure pulse. By using mitochondria as fiducial markers it was observed that this minimal magnitude of stimulus required to activate the ALM would result in an average strain of 3% strain at steady-state. This degree of strain was present regardless of mutations to the ECM encoding genes *mec-1*, *mec-5*, and *him-4*, the first two of which are necessary for touch sensation and will be discussed in the following section. The *him-4* gene is necessary for the formation of the fibrous organelles, which are similar to vertebrate hemidesmosomes. These interesting structures serve as a type of mechanical linkage or scaffolding via filaments that interface with transmembrane proteins and form cell-cell contacts [82]. In the case of the TRNs, this is between the plasma membrane of the TRN and the engulfing hypodermal cells. This finding suggests that these adhesions to the ECM or neighboring cells do not limit the amount of mechanical energy delivered to the TRN, but it could also be explained by a limited resolution of the device.

The final measurement tool whose development requires mention here is the Woodpecker system that allowed researchers to apply commanded indentations to the whole animal while recording electrical signals from the TRN and imaging in fluorescence (Figure 1.8.C). This open-loop system

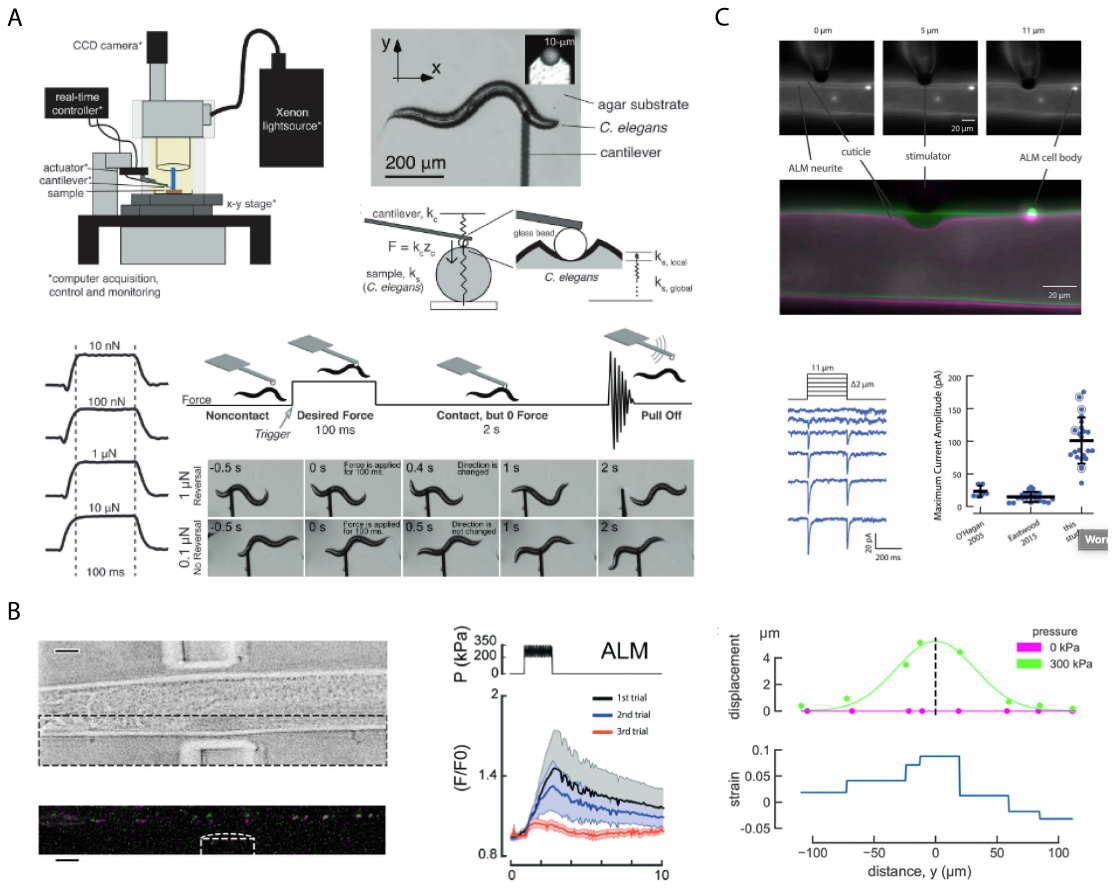


Figure 1.8: **Tools for assaying touch**. A. HAWK system uses piezoresistive cantilevers in a force-clamp system to apply mechanical stimuli to freely roaming worms. Reprinted from [20] with permission from Oxford University Press. B. Pneumatic stimulation of trapped worms combined with fluorescence imaging shows calcium transients in response to a buzz stimulus and strain between mitochondria during static deflection. Reprinted from [73, 81] with permission from American Society for Cell Biology and The Royal Society of Chemistry, respectively. C. Woodpecker applies open-loop commanded indentations to the TRN during fluorescence imaging and patch clamp recordings at faster speeds than previous experimental methods. Reprinted from [61] with permission from Rockefeller University Press.

also had higher movement speed capabilities that allowed for the first electrophysiological recordings of MRCs activated by faster and deeper mechanical stimuli. The peak MRCs elicited in these experiments were roughly four-fold higher, on average. In these studies the researchers also tested for an effect of probe position during stimulation with regards to the TRN cell body. Stimuli were applied either anterior to the cell body, above the neurite, or posterior in the region outside the TRN's purported receptive field. They saw a significant reduction in both peak MRC amplitudes in the posterior stimulations, as expected, but this effect was more profound for the MRCs at the removal of the stimulus at slower speeds. Another unexpected finding from this study was that the currents elicited did not saturate in magnitude as previously believed. MRC peak amplitude linearly increased with indentation depths for the entire commendable range. This is certainly a result of the technical limitations of the previous experimental method that could not achieve similar indentations in the same period of time due to their slower activation speeds. Conversely, measurements of the time required to reach the half-maximal MRC amplitude reached a lower asymptote of about 1 ms at roughly 6 μm indentation depth. Simulations applied anterior to the cell body resulted in progressively increasing decay time constant, meaning it took longer for the MRC current to decrease in value at larger indentations. These findings give important insights into the relationship between mechanical stimuli and channel kinetics.

1.5 Discussion

Mechanosensation in *C. elegans* is arguably the most well characterized of all multicellular organisms. Considerable efforts have contributed to building a comprehensive understanding of how touch is converted into neural impulse within milliseconds. Forward genetics screens completed over multiple decades have identified 16 genes required for mechanosensation. These genes encode a variety of molecules from transcription factors to ion channel subunits and microtubules proteins. Central to this sensory process is the MEC-4 channel subunit that is a member of the highly conserved DEG/ENaC family known to play key roles in health and disease vertebrates as well as invertebrates [53]. And yet, for as much as we understand about this system, there remain many unanswered questions. In particular, the question of greatest interest to this thesis is what are

the factors that deliver MEC-4 to its position within the plasma membrane, distributed along the neurite that allows for wild type mechanosensation? Only a subset of the MeT channels that are responsible for somatosensation in mammals have been identified, and even less is known about their localization patterns in mechanoreceptors. Thus, the nematode *C. elegans* provides an ideal system for uncovering these factors.

1.5.1 Contributions of this thesis

The distribution of MEC-4 puncta observed along the length of the neurite is the result of several underlying biological mechanisms, few of which have been identified. Previous studies could not address these more nuanced questions with regards to the determinants of channel positioning as the genetic tools available resulted in a blunt effect, eliminating nearly all puncta. At the start of this dissertation work it was hypothesized that isolated TRNs, cultured outside of the animal from dissociated embryonic cells, would not display MEC-4 puncta as they do *in vivo*. Against this prediction, one study used MEC-4-specific antibodies [23] to detect MEC-4 protein in dissociated and cultured TRNs and concluded that MEC-4 was present in the plasma membrane [60, 83]. This finding raised the possibility of using cultured TRNs could be used to directly measure channel gating. The limited nature of this prior dataset motivated me to look closely at MEC-4 expression and position in cultured TRNs, which became the central focus of this thesis. Initial evidence that cultured TRNs might retain mechanosensitivity came from unprecedented combined atomic force microscopy experiments that a calcium transient could, indeed, be evoked from a cultured TRN. This inspired experiments to quantify the distribution patterns of MEC-4 in isolated TRNs. However, these efforts were limited by the significant amount of heterogeneity seen in *C. elegans* cell cultures.

To overcome these challenges I took an engineering approach and developed tools to improve our measurement resolution by lowering the noise floor of the cell culture system. To do this I investigated the feasibility of reducing heterogeneity in two ways. The first is in the form of reducing the number of cell types present in cell cultures. This work is described in Chapter 2. The second is in the form of using protein micropatterning to restrict TRN morphology, as detailed in Chapter 3.

Finally, I developed high-throughput image analysis techniques that reduced the user bias that contributes to measurement error and bias. I use these tools to evaluate MEC-4 distribution and conduct an unparalleled quantification of hundreds of cultured neurons in multiple genetic backgrounds and conditions. This work is presented in Chapter 4 and suggests that potential aspects of MeT channel localization may indeed be cell autonomous. These findings have inspired a new area of research aimed at understanding the regulatory program that maintains channel distribution along the neurite. In the Conclusion, I provide a detailed discussion of the suggested future directions for this work.

Chapter 2

Computational and genetic approaches for reducing cell-type heterogeneity

2.1 Introduction

Uncovering the biophysical mechanisms that enable the rapid conversion of mechanical stimuli into neural impulses benefits from an *in vitro* approach that provides direct access to the molecules and cells of interest. By working with single cells outside of their native environment, researchers can precisely engineer the cellular environment in ways not practically possible *in vivo* making key scientific questions more tractable. For example, to date we do not know which aspects of mechanosensation may be cell-autonomous and which require surrounding tissues. Additionally, a complete understanding of cellular mechanics requires data on how the single cell responds to stress and strain independent of surrounding tissue. While cultures of embryonic cells from *C. elegans* have been established as a powerful method for conducting parallel *in vitro-in vivo* analysis [24, 35], their widespread use is limited by the low signal-to-noise system properties [84]. A prominent source of variation or noise in this system is the heterogeneity that is an unavoidable outcome of procuring single cells from dissociated embryos. This makes it technically difficult to conduct precise measurements and reproducible experiments. The objective of the work presented here was to reduce the heterogeneity of cell types by enriching for genetically-tagged cell types.

Although fluorescence-activated cell sorting (FACS) is clearly a very powerful tool for enriching a heterogeneous population of cells for further characterization (a detailed discussion of its use is provided in Chapter 1), utilizing this method for analyzing *C. elegans* and enriching dissociated embryonic cells has considerable challenges. During FACS, cells are suspended in solution and streamed through a fluidic system, passing at least one laser (the number of lasers varies with equipment models). When a large enough object passes by this laser, it activates the measurement of several parameters regarding this object including fluorescence intensity and proxies for particle size (front and side scatter). The cells are subsequently redistributed into collection tubes based on whether or not their measured values fall within operator-specified "gates." For example, a FACS operator may set a threshold on minimum particle size and fluorescence intensity in the green channel so that only large enough and bright enough particles are saved. This approach works particularly well for large cells that are easily discernible from debris, as well as those with very bright fluorescence expression levels. However, for small cells such as those from *C. elegans* (cell bodies are 2-5 μm in diameter), separating cells from small debris is a non-trivial task. The process is also limited when gene expression is low and thus fluorescence intensity from reporters is relatively dim. Circumventing these challenges requires investment into operator expertise and process optimization. In some cases, a more accessible approach is to simply image the cells in question and count the number that express the reporter of interest. This has been demonstrated in several *C. elegans* papers where dopaminergic (DA) neurons from various strains expressing reporters under uncharacterized genes were stained with DA antibodies and then imaged in a second fluorescent channel [85–87]. However, this approach is clearly limited by throughput and remains subject to error as results could potentially vary based on imaging conditions.

FACS experiments are useful beyond the sorting and collecting of cells. As the cells pass various detectors, information about the cell's size and fluorescence intensity in multiple channels is collected and stored. This can quickly become an unwieldy data set as each cell may have roughly 14 associated measurements and there will easily be hundreds to thousands of cells measured. Until recently the only tools for analyzing this data were proprietary software packages where users were most often required to draw manual boundaries around hypothesized clusters. As the software packages had no means for visualizing more than two parameters at a time, the process for

identifying clusters in more than two dimensions was tedious and time consuming. In the past few years, a number of groups have developed open-source software libraries for analyzing FACS data saved in the generic file format. With these libraries, users can write custom code to quickly parse large data sets and conduct quantitative profiling of population wide fluorescence expression. Such an approach would allow for screening hundreds of thousands of cells for changes in fluorescence, far exceeding the throughput of traditional single cell imaging methods. By computationally, rather than graphically, analyzing data sets can easily identify clusters in multiple dimensions using existing algorithms. This could potentially help with identifying cells that show weak fluorescence expression under single copy endogenous promoters. Since FACS machines require significant operator expertise, there is considerable motivation to use these computational tools to measure changes in population wide fluorescence expression as measured by a simple bench-top bioanalyzer that does not collect cells.

Seeking to overcome these limitations, we asked if bioinformatics tools could be used as a reduced-bias approach to measuring changes in population wide fluorescence expression in *C. elegans* embryonic dissociations. We hypothesized that we could us open source flow cytometry libraries to measure cell-specific enrichment of for *C. elegans* cells expressing genetically-encoded antibiotic resistance genes such as the widely-used puromycin resistance gene. To test this hypothesis, I refined the published cell culture methods to generate large populations of isolated cells. I then developed a protocol for treating cells dissociated from a strain expressing the puromycin resistance gene (puroR) under a TRN specific. The expected result is that TRNs would be spared from the ability of puromycin to kill eukaryotic cells in culture. To quantify TRN enrichment, I developed a set of computational tools in the language R to parse tens of thousands of single cell measurements collected on a bench-top flow analyzer. Pilot experiments used transgenic animals co-expressing a fluorescent reporter and the puroR gene under different, TRN-specific promoters. Next, we asked if we could improve the technique by expressing both genes (puroR and fluorophore) under a single cell-specific promoter sequence. Following up on this idea, I generated multiple transgenes having the base structure of promoter::puroR::SL2::fluorophore for use in a flexible CRISPR/Cas9 vector for single copy gene insertion into wild type animals. Dissociating embryos from these animals and running them through my puromycin treatment protocol and computational work flow,

I found that a single copy of the transgene generated fluorescent signals that were below the detection limits of the flow-based cell analyzer. However, control experiments with cells expressing multiple copies of a fluorescent reporter under the TRN specific *mec-17* promoter showed that the computational approach could clearly identify the subpopulation of cells using similar approaches. Together, these findings indicate that cell-specific antibiotic resistance may be a viable approach to reducing cell-type heterogeneity in cultures from *C. elegans*, and that computational tools can provide a reduce-biased analysis of cell culture population statistics on a much larger scale than previously achieved.

2.2 Results

2.2.1 Embryo collection, dissociation, and cell culture

The embryonic dissociation procedure has a reputation within the worm community for being technically challenging, published protocols [88, 89] have flattened the learning curve and made the procedure more accessible. The process begins with growing worms to adulthood and until a few hours after egg laying has begun to collect the worms by rinsing them off of the agar plates and into a collection tube (Figure 2.1.A(1)). Once collected, well-fed worms require no more than 10 minutes of incubation with a highly basic hypochlorite solution to lyse the worm bodies, releasing embryos. The embryos are rinsed and separated from the lysis debris using a sucrose gradient (Fig Figure 2.1.A(2)), and then transferred to chitinase for eggshell digestion (Figure 2.1.A(3)). Once the eggshell has been fully degraded, the embryos are mechanically dissociated by syringe aspiration (Figure 2.1.A(4)). As the cells are significantly smaller in size than those typically used in cell culture studies, approximating cell density with a hemocytometer can be challenging and will require optimization. Only the largest of cells are easily detected, while most cells appear as debris since they fall out of the focal plane (Figure 2.1.F). Following filtration to remove large debris, the cells are seeded onto cell culture substrates that have been treated with peanut agglutinin (PNA; Figure 2.1.A(5)). This entire procedure can easily be completed within two and a half hours for a single population of worms, or three hours for two populations (i.e., preparing two strains of distinct genetic backgrounds in parallel). With practice these cultures can be prepared using only aseptic

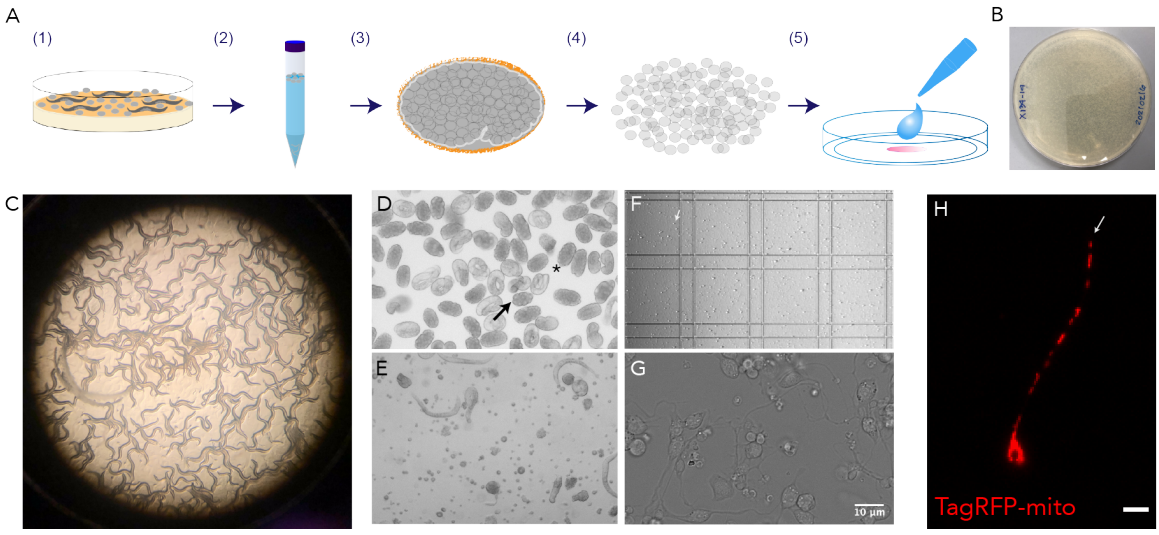


Figure 2.1: **Characteristics of isolated *C. elegans* embryonic cell cultures**. A) Embryonic dissociation procedure. 1. Animals are grown until gravid and washed off of plates for lysis. 2. Embryos are separated from lysis debris via sucrose gradient. 3. Incubation with chitinase degrades eggshell. 4. Embryos are dissociated by syringe aspiration. 5. Filtered dissociations are seeded onto cell culture substrates. B) Representative plate full with tens of thousands of gravid animals. C) Magnified view of plate with worms prior to collection. D) Embryos following chitinase digestion. Arrow indicates furrow in embryo cluster, indicating egg shell absence. Asterisk highlights pretzel stage larvae without eggshell. E) Mix of cells and partially dissociated embryos and newly hatched larvae. Asterisk highlights a larva that will be filtered out. F) Isolated cells following filtration as visualized in a hemocytometer. Arrow points to largest of isolated cells. G) Seeded cells after four days in culture. H) TRN expressing mitochondrial marker. Arrow points to distal most mitochondrion.

technique on a bench-top without a tissue culture hood without contamination. (Thus using a tissue culture hood is desireable, but not essential.) The cells are stable at standard room temperature and by using culture media that maintains pH without bicarbonate-based buffers there is no need for an incubator equipped with carbon dioxide regulation. Figure 2.1.G is a representative image of cells that have been in culture for four days without media changes, demonstrating that the cultures are not expanding. Cells can be cultured from nearly every genetic background, allowing for live cell imaging of fluorescent markers, such as the TagRFP-mito marker shown in Figure 2.1.H (worm strain provided courtesy of [90]). Again, as the cells thrive at moderate room temperatures and do not require carbon dioxide incubation, live cell imaging can easily be performed without the need

for specialized microscopy incubators.

A commonly cited issue when preparing these cultures is a low cell yield following dissociation, which can be caused by a number of factors. The easiest way to address this problem is to start with somewhere on the order of 25-50 thousand gravid animals. The easiest way to reach this goal is to grow worms on enriched peptone plates seeded with NA22 *E. coli* bacteria. This bacteria, when supplied with sufficient peptone, will produce a thick bacterial lawn that can support a higher density of worms than the more commonly used OP50 *E. coli* bacteria (Fig 2.1.B). Low cell yield can also occur if the lysis procedure is carried out for too long as the hypochlorite solution will damage the embryos. Extended lysis is only necessary when animals are starved, therefore it is critical to ensure an optimal number of worms per plate. Figure 2.1.C shows a plate with the maximal number of worm density for an NA22 enriched peptone plate. There is a trace amount of food left on the plates, as evidenced by the worm tracks in the bacterial lawn, and the worm's digestive tracts appear optically dense. Optimizing chitinase digestion time can also improve total cell yield, as excess digestion will also damage cells. *C. elegans* embryos are surrounded by two protective layers that make embryo dissociations technically challenging. The first is a thick, nearly impenetrable shell made of chitin [91], followed by a second protective layer called the vitelline membrane which can only be mechanically stripped away before the embryo can be dissociated into single cells. To do this the embryos must be passed through a narrow orifice following chitinase treatment. Digestion treatment will require a minimum amount of time (45 minutes), but should be checked regularly beyond this time point to avoid over-treatment as it will damage the cells and also reduce yield. Experimenters can visually inspect the solution at 10x magnification, checking if embryo cell clusters show visible furrows or indentations, indicating total egg shell digestion (Figure 2.1.D). The mechanical dissociation process should also be frequently checked to ensure that the cells are not excessively handled. Figure 2.1.E shows a representative checkpoint where mechanical dissociation is halfway complete. As contamination by environmental microbes is always a challenge to eukaryotic cell culture, additional caution when using aseptic technique is paramount to successful experiments. All surfaces should be treated thoroughly with 70% ethanol, once lysis is complete all solutions used must be sterile and samples handled with gloves. In practice I have found that wearing a face mask during preparation reduced the frequency of contamination, whereas using an

open flame to alter airflow had no observable effect.

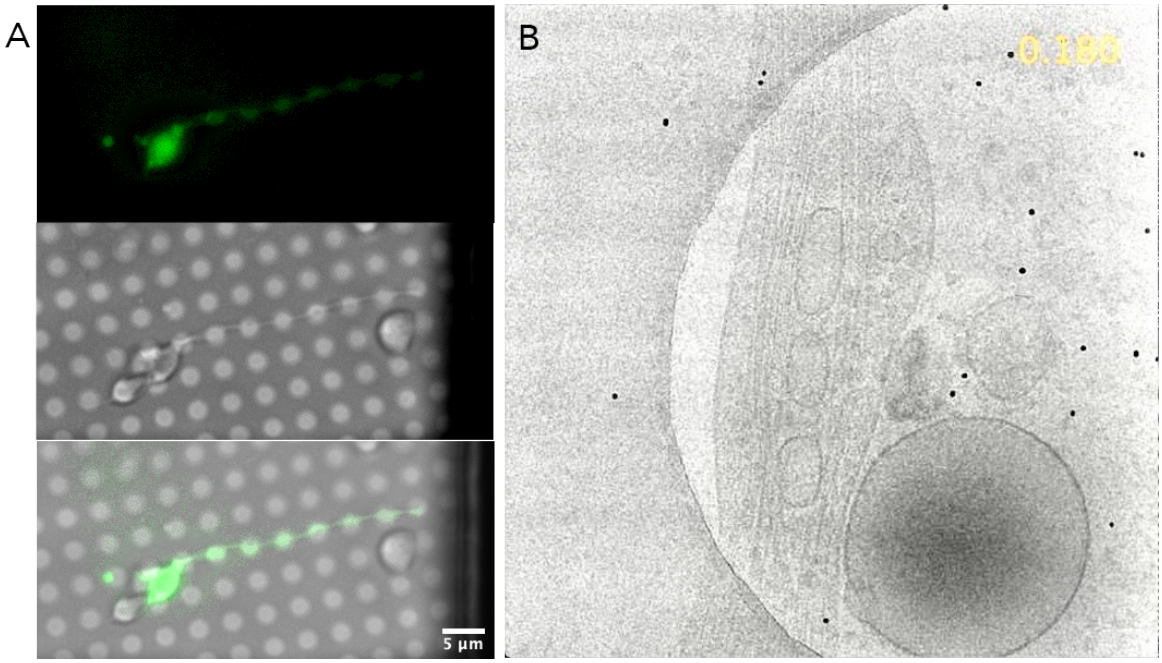


Figure 2.2: **Suitability of cultures for cryoET**. A) Isolated, GFP expressing TRN is attached to TEM grid that has been treated with PNA. B) Reconstructed tomogram of isolated TRN with fiducial markers (black dots) and what may be vesicles wedged between microtubules.

2.2.2 Applications of cultured TRNs

To better gauge the potential of this system to be used in various biological studies, I sought to characterize the suitability of these cells in two additional use cases: (1) cryo-electron tomography (cryo-ET), and (2) combined atomic force microscopy (AFM) and calcium imaging. The first set of experiments are motivated by the need to learn more about the structural organization of mechanoreceptor neurons so as to understand how mechanosensory functions. Cryo-ET is a powerful imaging tool capable of achieving nanometer scale resolution images of organelles in samples that have been rapidly frozen and thus are likely to have more accurate representations of cellular structures. Normally samples must undergo milling in order to reduce thickness, but we hypothesized that thanks to their small size, our isolated cells could be imaged while fully intact. Through a collaboration

with the Skiniotis Lab at Stanford, I demonstrated that TEM grids could be treated with peanut agglutinin (PNA), thus allowing the neurons to attach and extend processes as usual (Figure 2.2). As a result, the Skiniotis Lab was able to image these samples and generate cryoEM tomograms of one such neurite showing microtubules and vesicles that might be undergoing transport (Figure 2.2).

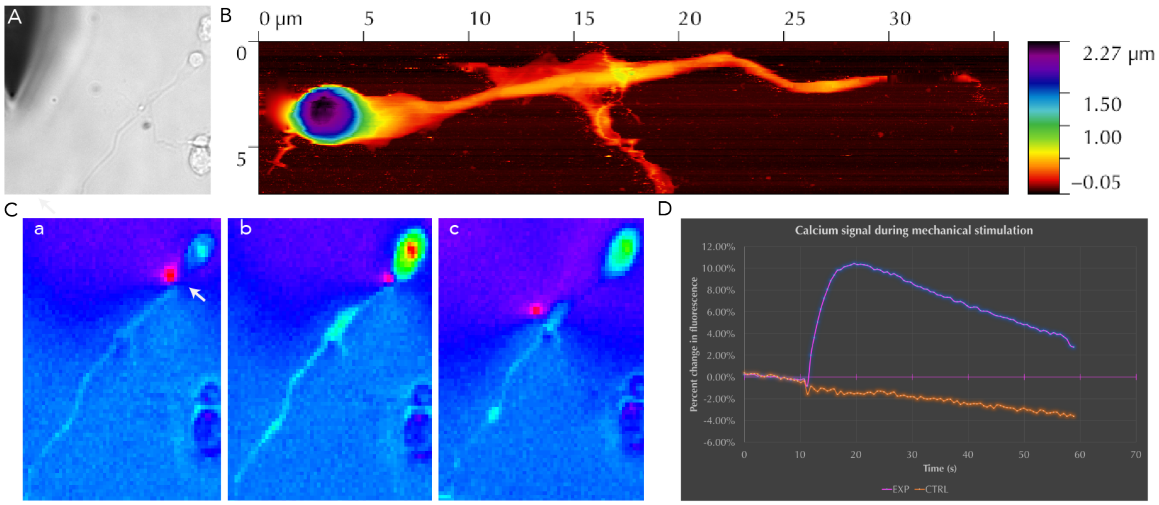


Figure 2.3: **Combined AFM and calcium imaging of isolated TRNs**. A) Brightfield image of isolated TRN and AFM probe. B) Isolated TRN topography as measured by BioAFM. C) Heat maps of calcium indicator fluorescence intensity before (a), during (b), and after (c) mechanical stimulation with AFM probe (pink dot). D) Quantified change in fluorescence intensity during mechanostimulation for region within neurite as compared to region away from cell.

The motivation to test if the neurons were suitable for combined AFM-calcium imaging originated from wanting to use these cells to measure the biophysics of mechano-electrical transduction channel gating. I wanted to know if the cultured touch receptor neurons (TRNs), expressing a calcium indicator in an optimized genetic background, would respond to mechanostimulation from an AFM probe. Previous efforts from others had attempted similar studies but suffered from heterogeneity of cells, cell-cell interactions interfering with stimulation, and noisy calcium indicator signals [84, 92]. To test this hypothesis I used a specialized AFM system that has a traditional scanning probe microscope head that sits above the sample, mounted to an inverted fluorescence microscope (Figure 2.3.A). Generating a topographical map of the identified TRN (Figure 2.3.B) enabled me to precisely specify where the AFM should touch the TRN for a mechanosensitivity

assay (Figure 2.3.C). Here I was able to visualize the TRN responding to being touched with the probe as an increase in fluorescence intensity that coincided with the repositioning of the probe tip from adjacent to the TRN (Figure 2.3.C(a)) to directly on top of the axon initial segment (Figure 2.3.C(b)). The probe tip was promptly removed and the calcium transient gradually decreased (Figure 2.3.C(c)). Using ImageJ, I traced the cell body and neurite of the target and calculated change in fluorescence intensity as normalized to baseline levels and plotted this value for each frame of the continuous recording. This quantification suggested a more than 10% increase in total fluorescence intensity as compared to a neighboring, unprobed cell, sharing some similarity to *in vivo* measurements from the parent strain [81] (Figure 2.3.D).

2.2.3 Tools for high-throughput analysis of dissociated *C. elegans* embryos

Bench-top flow cytometers, such as the BD Accuri bioanalyzer used in these experiments and depicted in Figure 2.4. They differ from FACS machines primarily in that samples are analyzed but not physically sorted and collected. Additionally, users are not able to manually adjust the detector gain as they normally would during a FACS experiment, and as such it can be more challenging to discriminate weakly fluorescent from baseline or auto-fluorescent events. However, in the absence of needing to tune the system prior to each experiment, each experiment requires substantially less setup time and data can be collected without expertise knowledge or extensive equipment training. As events are analyzed the data are transferred to proprietary software where users will typically interact with the events via a user interface, manually identifying populations of interest. Most often, the vendor provided software allows for data to be exported in the standard *.fcs file format, which can then be further processed using third party software or custom code utilizing one of the many flow cytometry codebases available in multiple programming languages. I hypothesized that I could use this approach as a basis for developing a fully automated, high-throughput classification algorithm to quantify population characteristics for isolated *C. elegans* embryonic cells. Such a classifier could provide a basis for using this cell culture system for high-throughput toxicity assays or to measure changes in population wide gene expression.

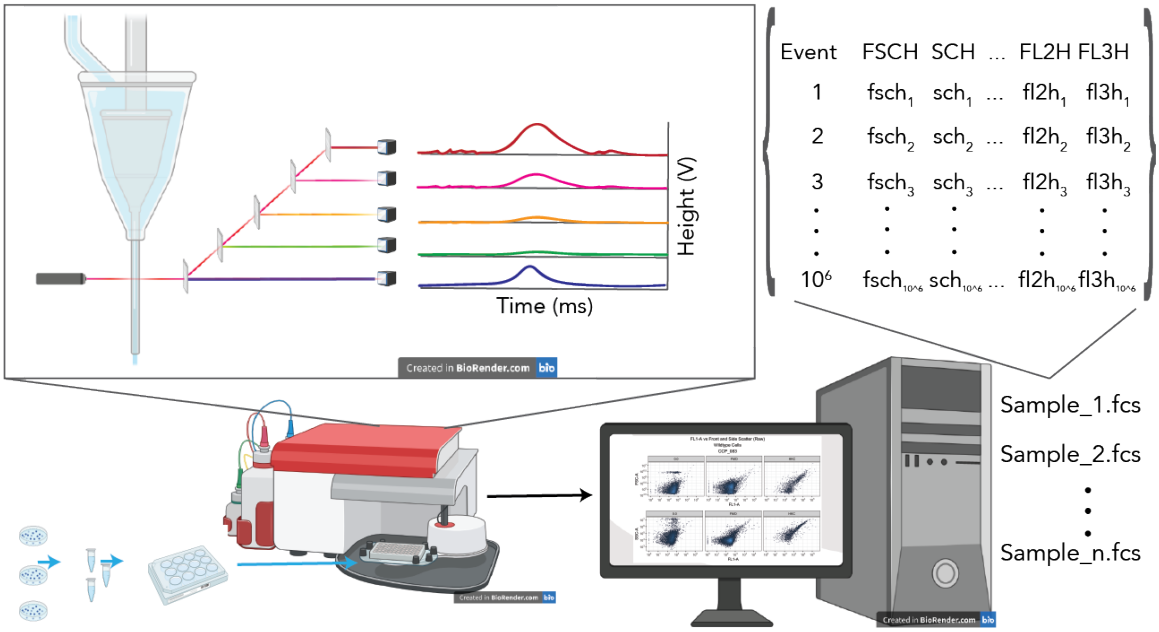


Figure 2.4: **BD Accuri system overview**. Samples are collected and mounted into the bioanalyzer which measures multiple parameters and sends data to computer for storage as .fcs files. Created with BioRender.com.

Before testing this hypothesis directly, I benchmarked the system performance as it pertained to my samples of interest. As described earlier, the small size of the *C. elegans* cells, dim fluorescence levels, and high auto-fluorescence levels of wild type cells warranted concern with regards to experiment feasibility. The BD Accuri C6 used in these experiments has the most basic set of features and was thus a perfect test-bed for developing a widely accessible algorithm (i.e., the developed algorithm could be used by experimenters even if they only have access to a simple model). The workflow for operating this tool is presented in 2.4, along with a graphical representation of the data configuration. The analyzer can accommodate flow rates up to 100 µL/min, and measure up to 1 million events per well, in four fluorescence channels (shown as green, orange, pink, and dark red lines in 2.4). For measuring fluorescence the device has two separate excitation lasers, 488 and 640 nm, which are optimal for the standard filter set used in these experiments (FL1 533/30 nm, FL2 585/40 nm, FL3> 670 nm, FL4 675/25nm). For simplicity, these

channels are referred to as green, orange, near red, and far red, respectively. Samples are prepared and mounted into the analyzer, which then generates a laminar stream of cells flowing past a set of lasers and detectors (only one laser is shown in 2.4). As the light filter bandwidths are partially overlapping, the potential for spectral overlap across the channels must be considered in developing an accurate classifier. The data from each event is transmitted to the accompanying computer terminal which runs the proprietary software, capturing data for each event as it is acquired. Users must then export this data into a standard *.fcs file using the software's built in function. For the purposes of the classifier developed here, each sample requires its own *.fcs file, containing only the events for that sample. These files contain additional experiment parameters, however most critical are the data collected from the FSC ("front scatter"), SSC ("side scatter"), FL1, FL2, FL3, and FL4. Front and side scatter refers to the pattern of light that is formed when an object passes the laser. In the absence of an object, the laser lands on the detector without disruption. However, when an object crosses the laser's path, the light is scattered and does not create the same pattern on the detector. Larger objects create more scatter, and thus FSC and SSC can be used as proxies for object size either as viewed from the front or from the side.

To benchmark the system's performance, I analyzed samples from unlabelled, wild type (N2) and *unc-119*::GFP(OH441 [93]) animals expressing GFP in all neurons. Cells were prepared from embryonic dissociations following the previously described dissociation procedure (Figure 2.1), as adopted from [88, 89]. Analysis was conducted immediately after the dissociation procedure using standard equipment procedures and default settings with flow rates of 35 µL/min and 16 µm core. After data collection, I used custom code that leveraged the previously published R packages "flowCore" [94] and "ggcyto" [95] to visualize the measured events. FlowCore provides methods for importing *.fcs files into the R environment as either individual "flow frame" objects containing a single *.fcs file, or compiling multiple files into a single "flow set" object. The objects contain all data collected and metadata associated with each event, such as front/side scatter and fluorescence values measured in each channel. The measured values can then be visualized as two-dimensional histograms with ggcyto, which is capable of taking either a single flow frame or flow set object as an input and plotting any of the channel values as aesthetics. The ggcyto method adheres to the grammar of graphics widely used in R's ggplot. Visualizing the flow cytometry data as 2D

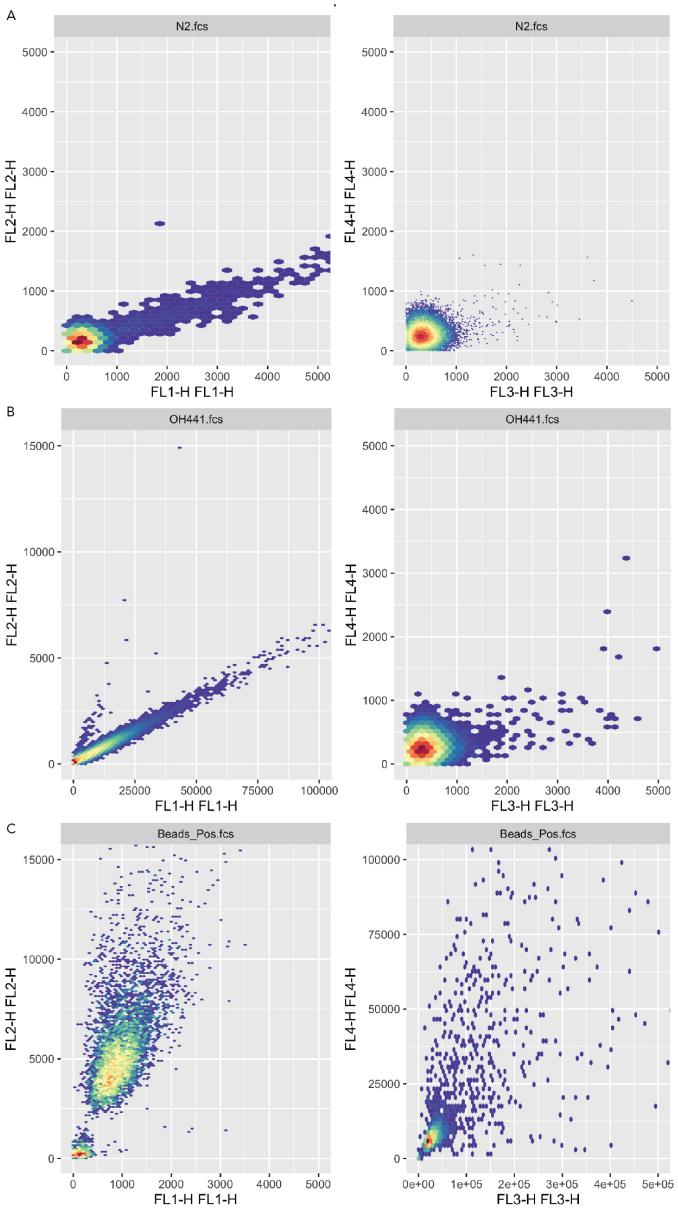


Figure 2.5: **Bioanalyzer benchmarking**. A-C: Two dimensional histograms showing FL2H versus FL1H, and FL3H versus FL4 for (A) wild type (N2), (B) *unc-119::GFP* (OH441), and (C) ArC reactive, stained compensation beads.

histograms with distinct fluorescent channels as orthogonal axes also allows the user to characterize the degree of fluorescence spillover between detectors. Spillover due to spectral overlap between filters and can lead to inaccuracies in classification (i.e., a green event in FL1 can appear as orange in FL2). This is a substantial challenge in flow cytometry experiments that typically requires substantial effort to correct in the form of compensation [96]. As expected, I found that the N2 cells showed high levels of auto-fluorescence primarily in the green (FL1) channel, spilling over into the orange (FL2), but neither of the red channels (FL3, FL4; Figure 2.5.A). The OH441 cells (predicted to be at least 75% GFP expressing [41], appeared much brighter in the green channel than the N2 cells, with similar spillover characteristics (Figure 2.5.B). These observations suggest that future studies might benefit from using red or far red fluorescent proteins as cell markers. For comparison I also analyzed commercially available 6 μm diameter polystyrene beads coated with amine groups, treated with an amine-reactive red dye. These beads are commonly used as fluorescence controls to calculate the spillover matrix necessary for compensating data. True to expectations, these beads appeared brightest in the near red channel (FL3), spilling over primarily into the orange channel (FL2), with slight spillover into the green channel (FL1; Figure 2.5.C). These results confirmed both that the cells could be accurately detected by this flow cytometry system and analyzed with code written in R using the packaged flowCore and ggcyto.

2.2.3.1 Live/dead classification of wild type cells from flow cytometry data

Prior to classifying events as cells that are either alive or dead, the data sets must be analyzed and processed for quality control following standard flow cytometry procedures [97, 98]. Data pre-processing begins with compensating data for spillover, achieved computationally by generating a "spillover matrix" which is then applied to the data set (treated as a linear operation applied to the data which is represented as 2D matrix of values) [94, 96]. To generate this spillover matrix a set of control data are used to measure the degree of fluorescence spillover between channels. FlowCore does this by reading in a flow set that contains at least one flow frame for each fluorescence channel and one additional frame for measuring front and side scatter relationships. FlowCore also has a simple, built-in method for compensating flow sets once this matrix is computed. To calculate this matrix, I prepared and analyzed samples of unlabelled, GFP-expressing, orange-fluorescent

protein expressing, and red compensation beads, using the FlowCore generated compensation matrix. Unfortunately, I observed that adding this compensation step to the algorithm reduced the data quality following this procedure. The compensation process resulted in many of the samples having negative fluorescence values and it seemed to increase the correlation between channels (Appendix A). Inspecting the spillover matrix, I found that, unlike the expected matrices described by [96], the computed result was not a symmetric matrix with values of one along the main diagonal. One possible explanation is that the high levels of auto-fluorescence that appear in channels FL1 and FL2 even for wild type cells led to a computational error. Such challenges with compensating auto-fluorescent cells have been previously described [99]. To test if not compensating the data would alter live/dead classification, I visually checked for spillover between FL3 and FL1 using bright red beads. As shown in Figure 2.6.A, even the highest levels of fluorescence intensity in the near red (FL3) channel do not correlate with increased brightness in the green (FL1) channel. Thus, for the purposes of this algorithm, I opted to not compensate data sets prior to classification.

Following compensation, flow cytometry data sets are typically "gated" to remove questionable samples such as debris, cell doublets, or clusters. To establish ground truth for object size, I analyzed commercially available polystyrene beads of three different sizes (2, 4, and 6 μm) commonly used for calibrating flow cytometry measurements. Front scatter is typically considered a more reliable measure of cell size, as the cells are deformable and are more likely to become elongated in the direction of flow, and thus I focused my analysis on generating a linear model to relate bead size to FSC-H measurements. Appendix A presents the details of the generated model, including the corresponding conversation factors used to predict FSC-H values given a size in microns. I developed the code such that the user specifies size boundaries in microns at the beginning of the analysis, these boundaries are automatically converted to FSC-H values using the linear model during analysis, and then used directly to gate data sets to remove samples below or above the size thresholds, lower and upper respectively. To identify potential statistical guides for setting size thresholds, I plotted the distribution FSC-H values for isolated cells but found the events were monotonically distributed (Appendix A). In the absence of visible FSC-H trends, I relied on AFM topography scans of isolated cells (Figure 2.6.B) to set a lower size bound, which suggested that the cell body for an isolated TRN is 3 μm in diameter. From working with the cells in culture, I have observed

dead or dying cells that are at least three times as large as TRN cell bodies. Wanting to ensure that size gates were sufficiently large to accommodate dying cells, I set an upper limit of 10 µm. Following standard guidelines for flow cytometry analysis, I analyzed front versus side scatter plots to test for the presence of doublets but did not observe discrete populations that could be easily gated, thus I omitted doublet gating from the algorithm.

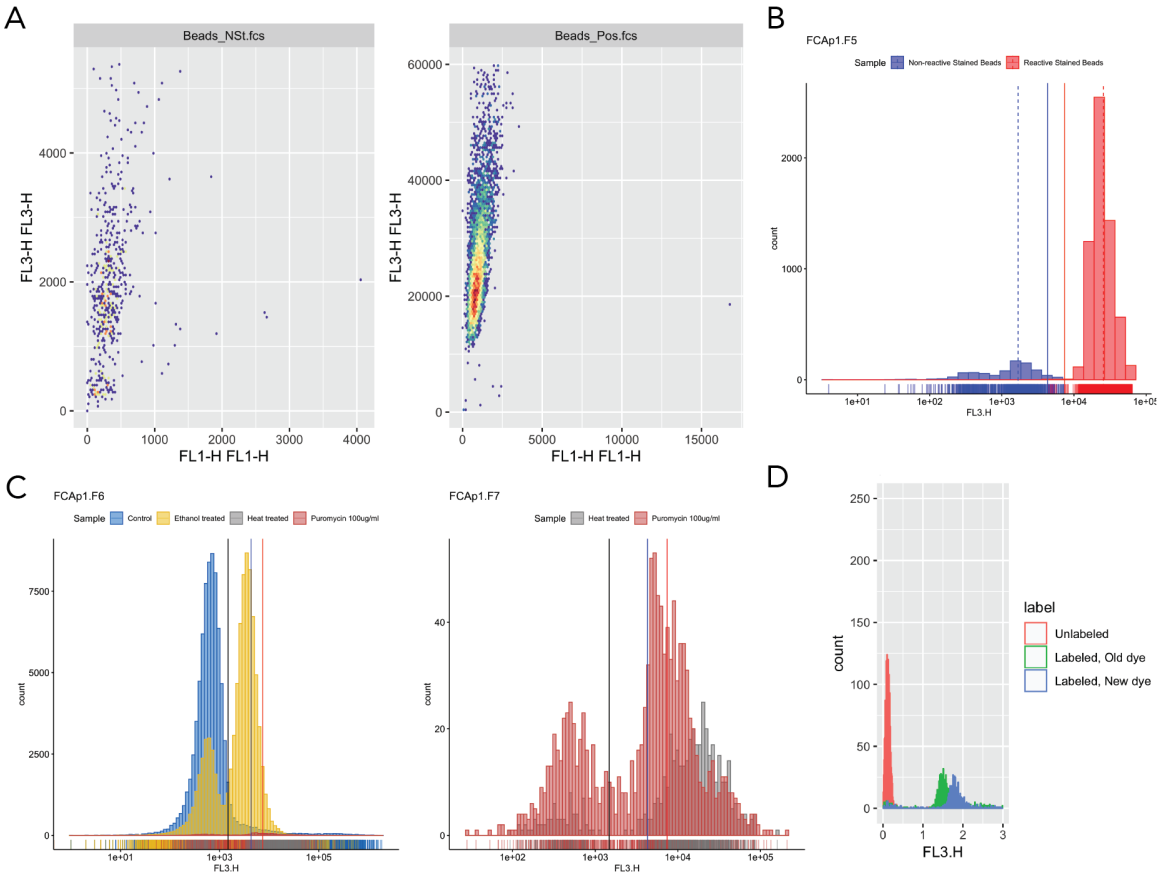


Figure 2.6: **Calibration of live/dead cutoffs from experimental training data sets**. A) Two dimensional histogram showing FL3H versus FL1H for uncoated (Beads_Neg) and amine coated (Beads_Pos) beads stained with ArC. B) FL3H histograms for beads shown in (A). Population means for 2-3 quartile subset are shown as dashed lines in blue (Beads_Neg) and red (Beads_Pos). Solid lines show calculated live/dead thresholds (blue and red, respectively). C) FL3H distribution for wild type (N2) cells with either no, ethanol, puromycin, or heat treatment. Solid black line marks the 30th percentile for puromycin treated, N2 cells stained with PI. D) Logicle transformed FL3H distributions for amine coated compensation beads either unstained, labeled with freshly prepared or week old ArC.

Fluorescent markers such as amine reactive dyes (ArC) or propidium iodide (PI) are frequently used in flow cytometry experiments to discriminate living from dead cells. When these dyes are able to bind amine groups or DNA, respectively, as is the case when the cell membrane becomes compromised, the cells appear intensely fluorescent. Prior to analyzing stained cells, I established ground truth for classification by analyzing polystyrene beads, with and without amine groups, stained with ArC. Figure 2.6.A shows 2D histograms for these beads in the near red (FL3) versus green (FL1) channels. Beads possessing an amine group coating (Beads_Pos.fcs) show a 10-fold increase in fluorescence intensity over beads not having this coating (Beads_NSt.fcs). In this analysis I also observed that these uncoated beads were further divided into two distinct clusters, the dimmer of which having red fluorescence intensities barely discernible from background levels. To determine live and dead cutoffs using these distributions, my algorithm calculates the 95% confidence interval for the uncoated beads (assuming a normal distribution), and uses this interval to set the upper "live" threshold (blue line, Figure 2.6.B). The algorithm then removes events from the coated bead sample using the "live" threshold, calculates the 95% confidence interval for the remaining population, and sets the lower bound of this interval as the "dead" threshold (red line, Figure 2.6.B). Population means for both data sets, post gating, are shown as dotted lines in blue and red, respectively.

To test if these thresholds could be used to accurately classify isolated *C. elegans* embryonic cells, I analyzed cells from wild types animals labeled with either ArC or PI. To generate training data sets that could be used to validate live/dead thresholds, I subjected cells to a lethal challenge with the goal of killing the majority of cells. Ethanol samples were incubated with 70% ethanol for thirty minutes immediately following dissociation, prior to ArC incubation. Cells expressing the puromycin resistance gene (puroR) were incubated with 100 ug/ml puromycin for 3 hours while rotating, then seeded on PNA treated substrates for 24 hrs, released, and then labeled with PI. Heat-killed cells were cultured on PNA-treated substrates for 24 hrs, released, placed in a heating block set to 65°C for 30 minutes, and then labeled with PI. Near-red (FL3) distributions for each of these samples, following size gating, are shown in Figure 2.6.C along with the live and dead thresholds previously calculated (blue and red lines, respectively). Unlike the ideal case of stained beads, the two distinct populations are overlapping at their tail ends. Evaluating the position of the live threshold against the brightest of the two FL3H populations for all lethally challenged samples it is

clear that these events are substantially dimmer than expected. Neither of these thresholds are set to accurately differentiate the hypothesized live and dead populations. To identify the point at which the two populations are distinct, I programmed the algorithm to quantify the 30th percentile for the puromycin-treated sample. This value is plotted as a black line on both graphs in Figure 2.6.C. As this value was more accurate in separating the two populations, I programmed the algorithm to use this value as the threshold between 'live' and 'dead' events. To further investigate possible explanations for why cells appeared dimmer than expected, I tested two batches of ArC against each other, again using amine coated beads. These batches were prepared several days apart but the beads were stained and analyzed at the same time. These results are shown on a logicle transformed axis in Figure 2.6.D and suggest that achieving optimal ArC brightness in staining requires careful protocol optimization as suggested by others [100].

2.2.3.2 Strategies to reduce cell type heterogeneity

2.2.3.3 Linear modeling of auto-fluorescence for improved GFP(+) classification

The primary goal of this work was to develop a classifier that could identify green fluorescent protein (GFP) expressing cells. Having established that the algorithm could filter flow cytometry data based on size and living vs. dead, I then asked if I could extend this classifier's capabilities to also identify isolated *C. elegans* embryonic cells expressing GFP. To develop an accurate classifier, I prepared and analyzed samples from a strain of *C. elegans* expressing GFP under the pan-neuronal *unc-119* promoter [93]. Although only a subset of embryonic cells from dissociations are predicted to express the *unc-119* gene (75% [41], I hypothesized that with proper pre-processing I could use these cells to establish ground truth for GFP classification. Visualizing the orange (FL2) versus green (FL1) fluorescence intensity of the isolated cells as 2D histograms I observed two distinct populations, separated by the correlation of intensities between the two channels (Figure 2.7.A, OH441.fcs). To test if the smaller of the two populations were likely to be the highly auto-fluorescent, non-GFP subset, as was seen in [101], I developed a linear model relating the fluorescence intensities in FL2 versus FL1 for wild type cells. The fit for this model is shown in Figure 2.6.A, N2.fcs, blue line. I then programmed the classifier to use this linear model to calculate the

vertices for two polygon gating objects that would divide the 2D FL2 FL1 space into two regions: Autofluorescent (AF) and NAF (non-Autofluorescent). I predicted that if I filtered the events within the auto-fluorescent cells OH441 population it would improve the classifier's ability to confidently score GFP(+) events. Following the removal of events within the auto-fluorescent bounds, I programmed the algorithm to calculate the 1.5-percentile of the remaining OH441 subset and use this value as the threshold for GFP classification. This cutoff is shown as a vertical green line in Figure 2.7.B, clearly indicating the divide between wild type, non-GFP expressing cells, and the OH441 population.

Although distinguishing GFP(+) from wild type cells is relatively straightforward in unstained samples, it remained unclear if staining with either ArC or PI would impair the classification process. To test if the developed NAF polygon gate could accurately exclude stained wild type cells, I plotted the polygon gate against stained N2 samples. Wanting to be certain that the results wouldn't vary with cell viability, I tested the gate against control samples as well as those challenged with either puromycin or heat, as described earlier. In doing this I observed that the gate excluded nearly all events in every case (Figure 2.7.C), although the margin of error was smallest for controls. This was expected as the boundary was initially drawn using unstained control cells. Conversely, I performed the same testing using stained OH441 cells that had either received no additional treatment (08_OH441_0.0ugml-1.fcs) or had been treated with 100 ug/ml of puromycin following the same methods used to prepare wild type cells. As seen in Figure 2.7.C, lower panels, the NAF gate accurately bound the majority of the OH441 samples. Interestingly, I also observed a change in the distribution of events in the AF region (i.e., the lack of a distinct 2nd population). By comparing these events to those of stained wild type cells I concluded that this redistribution of events was to be expected following staining. Finally, I validated these results by analyzing the distribution of stained OH441 cells in the green (FL1) channel along with the predicted GFP threshold before and after auto-fluorescence gating (Figure 2.7.D). In both panels distributions for control, heat treated, and puromycin treated cells are shown along with the same vertical line indicating GFP threshold from panel B. Before AF gating all distributions are nearly centered about the threshold. For control cells not treated with puromycin or heat, the threshold nearly intersects with the shallow valley between the two population peaks. Following AF gating there is little to no overlap between these

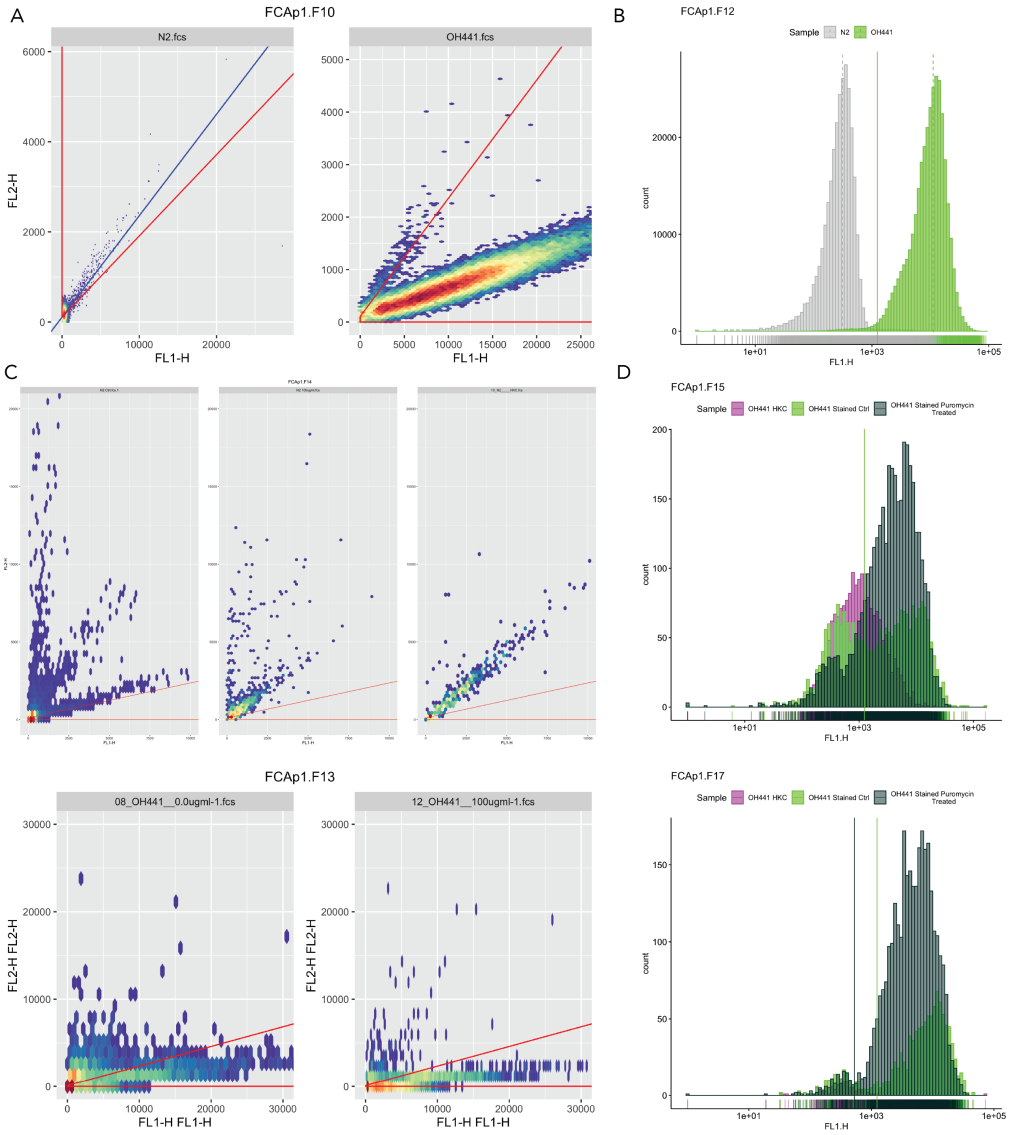


Figure 2.7: **Calibration of GFP(+/-) from experimental training data sets**. A and C: Two dimensional histograms showing FL2H versus FL1H for N2 and OH441 cells in various conditions. A) Unstained N2 and OH441 cells with linear model of autofluorescence (blue) and AF/NAF polygon gates derived from this model (red). B) FL1H distribution for cells shown in (A) with vertical green line (same for all panels) indicating the 1.5th percentile for OH441 cells. C) 2D Histograms for N2 (upper row) and OH441 (lower row) cells either untreated, puromycin or heat treated and stained with PI. D) FL1H distributions for OH441 cells shown in (C) before and after AF gating. Dark green line marks the 60th percentile puromycin treated sample.

populations and the GFP threshold intersects with the lower tail of the brighter and larger population. These results indicate that automated AF gating could improve the resolution of the classifier in differentiating GFP(+) from auto-fluorescent events.

2.2.4 Puromycin treatment and genetically-encoded puromycin-resistance to enrich for TRNs and other cell types

Having fully developed the classifier to identify and categorize cells based on fluorescence, I then asked how this system would perform when tasked with quantifying population wide changes in response to drug treatment. Antibiotic selection is a commonly used strategy for reducing a heterogeneous population of cells or organisms to a single genotype. Only those stably expressing the indicated antibiotic resistance transgene are able to withstand treatment with the matching antibiotic. Puromycin is one such antibiotic, known to effectively kill *C. elegans* not expressing the puromycin resistance transgene which encode puromycin (puroR) [102]. Our group previously developed a strain of animals expressing puroR under the TRN specific *mec-18* promoter (pgSi71), and crossed this strain with a previously developed animal expressing GFP under the TRN specific *mec-17* promoter (uIs31). We hypothesized that treating cells from these animals with puromycin could be used to enrich the cultures for TRNs. We further asked if the classifier developed in this work could quantify the changes in number of GFP(+) cells following puromycin treatment. If the puroR gene does protect isolated *C. elegans* embryonic cells from puromycin treatment we would expect to see an increase in the number of GFP(+) cells in treated groups up to the optimal drug concentration. We predicted that at high enough levels of puromycin even those cells expressing puroR would be unable to survive treatment and we would see successively lower levels of GFP(+) cells. To test these hypotheses I prepared and analyzed cells from two worm strains. The first, GN595, expresses both the puroR transgene and the GFP marker in TRNs. The second, TU2769, expresses the same GFP marker (uIs31) but does not have the puroR transgene. The cells were prepared as described previously, incubated with varying concentrations of puromycin, then seeded on poly-l-lysine coated culture substrates for 24 hrs. Following time in the culture, the cells were released from substrates using vigorous pipetting and labeled with PI prior to flow cytometry analysis.

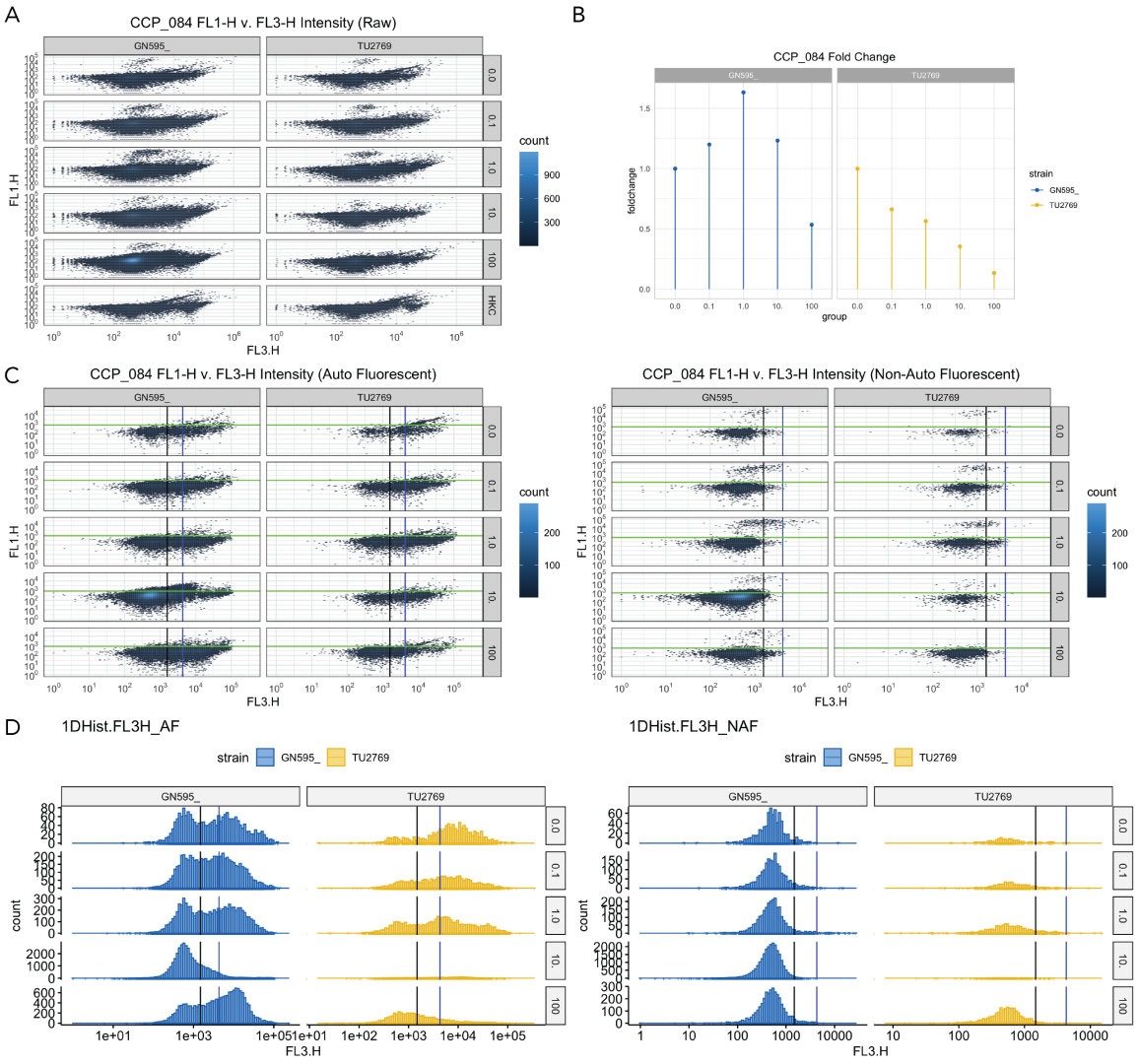


Figure 2.8: **Quantification of change in GFP(+) expression following puromycin treatment.** A and C: Two dimensional histograms for *mec-18p*::puroR;*mec-17p*::GFP (GN595) and *mec-17p*::GFP (TU2769) cells following treatment with varying concentrations of puromycin. C and D: Vertical black and blue lines correspond to live/dead cutoffs in Figure 2.7. A) Raw data for all samples plotted as a single flowSet. B) Quantified fold change in GFP(+) live cells following puromycin treatment. C) Debris-gated data divided into AF and NAF flowSets based on AF/NAF gating. D) FL3H Histograms of data presented in C showing live/dead populations.

To quantify fold changes in GFP(+) cells I processed *.fcs files from these experiments through

the fully automated classification system developed in this work. The details of this analysis are provided in Appendix A as compiled markdown files. The classifier first computes size gates, live/dead thresholds, AF/NAF gates, and GFP(+/-) cutoffs using the procedure and training data sets described earlier. Once these values and objects are stored in the R environment, the experimental data set is loaded and pre-processed for quality control. Two dimensional histograms for all samples for both strains are generated at each step, allowing the user to visualize intermediate data. Population statistics are calculated following each processing step and all intermediate data are saved to an analysis folder. In this process raw fluorescence and scatter intensities are converted to data frames which are then stored as *.csv files, enabling users to conduct manual analysis on data directly if desired (i.e., as opposed to working with the *.fcs file format). During quality control preprocessing I confirmed the presence of at least two populations in the raw data (Figure 2.8.A), the smaller of which being brighter in the green (FL1) channel than the larger cluster. Since TRNs are estimated to account for <1% of all isolated *C. elegans* embryonic cells [42], this relative size of this cluster seemed appropriate for the samples in question. Surprisingly, I did not observe a clear separation of live/dead populations in the near red (FL3) as expected from training data sets. Following the quality control documentation process, the classifier gates the single flowSet object containing all *.fcs files for debris and cell clumps. Next, the single flowSet is divided into two smaller sets using the AF and NAF polygon gates described earlier. The data for these two sets is again visualized as 2D histograms, shown in Figure Figure 2.8.C, along with the previously computed thresholds. The horizontal green line is the GFP(+) cutoff, the vertical black line is the live/dead cutoff computed from stained wild type cells, and the vertical blue line is the same cutoff but determined from ArC reactive and stained beads.

Inspecting the 2D histograms in Figure 2.8.C shows that AF gating accurately divides the populations such that samples with high correlation between FL1 and FL3 (the region defined by the blue and green lines) are removed from the NAF flowSet. From analyzing training data sets it is most likely that these are dead, non-GFP expressing cells that appear bright green because of their combined auto-fluorescence and bright red labeling. Visual inspection of the NAF flowSet histograms in panel C shows that following AF gating, the remaining populations show a monotonic relationship between FL1 and FL3. To improve visibility of potentially distinct live and dead populations I

programmed the classifier to plot one dimensional histograms of FL3H intensities for all samples, along with the same predicted live/dead thresholds described earlier (black and blue lines, Figure 2.8.D). As expected from the 2D histograms, the live and dead populations are clearly overlapping with very little separation between peaks in most cases. Interestingly, the AF flowSet shows large populations for both live and dead regions, where all NAF samples have only a main population in the live region. Based on the previously established guidelines, I programmed the classifier to count all events in the AF flowSet as being non-GFP, but only those events above the GFP threshold in the NAF flowSet were counted as GFP(+). For both flowSets all events were counted as live or dead depending on their position relative to the live/dead cutoff indicated with the black line (i.e., the cutoff derived from training data sets of lethally challenged and PI stained wild type cells). Finally, to quantify changes in live GFP(+) events for all samples, I programmed the algorithm to quantify the percentage of live GFP(+) cells for each sample (100*live GFP(+)/all live events, AF and NAF combined). Then, the algorithm divides the sample's percentage of live GFP(+) cells by this value for the control (0.0 ug/ml) sample. The results of this quantification are shown in Figure 2.8.B as a fold change. We see that, as expected, the samples with no puromycin have a fold change of one. Samples from the worm strain expressing the puroR gene show an increase in fold change up to 1 ug/ml and subsequently decline with increasing puromycin concentration. As predicted, cells from animals not expressing the puromycin gene showed a steady decline in fold change with increasing puromycin concentration.

2.2.4.1 Benchmarking classifier performance in a low signal-to-noise system

The initial motivation of this work was to develop a system that would allow users to easily and rapidly quantify population wide gene expression using high-throughput analysis of fluorescent reporters. Having established that the developed classifier could accurately identify GFP(+) cells in a well characterized genetic background (i.e., GFP expression under the very well documented *mec-17* promoter), we sought to test this approach with promoters whose expression levels were less understood. Specifically, we asked if we could measure changes in population-wide fluorescence expression of cells also expressing the puroR transgene, but this time under the same promoter as the fluorescent reporter. Such an approach would allow us to directly relate the puroR with fluorescence

expression, unlike the previous case where the puroR transgene was expressed under a separate promoter. To test this idea I leveraged an existing CRISPR/Cas9 flexible vector, developed by A. Das and based on the work described in [103], that allowed me to insert the transgene of interest into MosSCI site ttTi4348 on chromosome I. Using standard molecular biology techniques, I developed three transgenes, all following the structure presented in Figure 2.9.A. Under the expression of each promoter was the coding sequence for the puroR transgene, followed by the splice leader sl2, and either the mNeonGreen (mNG) or worm scarlet (wSc) fluorophore gene. We hypothesized that this transgene structure would yield a one-to-one relationship between puroR and fluorescence intensity levels. I produced three strains of animals using this approach, each with a different promoter in the transgene sequence. Using the well characterized mec-17p we confirmed that positioning the fluorescence gene after sl2 would not appreciably alter fluorescence intensity, as seen in Figure 2.9.G. Here, the TRNs are clearly visible as bright cell bodies having easily identified processes. While fluorescence expression levels for the mec-17p were sufficiently high enough to easily identify the cells of interest in adult animals, this was not as clear for the other two promoters used (*vha-7p* and *cmk-1p*, respectively). Panels C and E of Figure 2.9 show whole animal images for the transgenic strains which both express mNG under their respective promoters. In each case it is difficult to distinguish bonafide fluorescence in the expected cells (hypodermal and pan-neuronal, respectively). To verify that these animals did indeed possess the transgene in the correct location I conducted a PCR test on worm lysate prepared from the transgenic animals. In each of the three samples tested per worm strain I detected the presence of a strong band at the predicted location (Figure 2.9.B). To test if the fluorescence levels would be easier to detect in isolated embryonic cells, I dissociated cells from both strains and imaged these samples at 60x magnification with an inverted fluorescence microscope equipped with a standard GFP excitation/emission filter set. Representative images for these samples are shown in Figure 2.9.D, F. As a final test, I used flow cytometry as described earlier to analyze entire samples of cells from the transgenic animals. Using the developed classifier I tested if the samples contained appreciable levels of GFP(+) cells. As seen in Figure 2.9.H, neither strain showed GFP(+) clusters in the predicted location (i.e., above the GFP(+) classifier cutoff, as compared to OH441 animals). Although both strains do have a subset of events to the right of the blue line (live threshold calculated from ArC reactive and stained beads) and above the

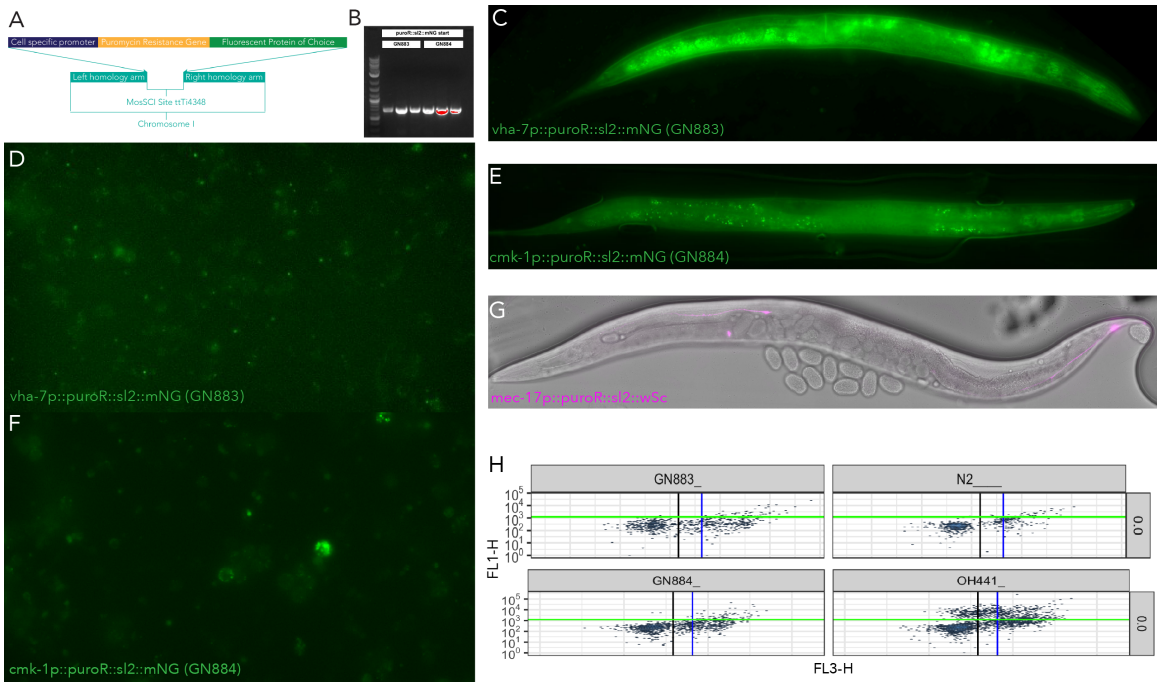


Figure 2.9: **Analysis of transgenic animals expressing puroR and fluorescent markers under a single promoter**. A) Flexible vector design for CRISPR/Cas9 single copy insertion of puroR and fluorescent reporter under a single promoter. B) PCR confirmation of transgene insertion for *vha-7p*::puroR::sl2::mNG (GN883) and *cmk-1p*::puroR::sl2::mNG (GN884). C) GN883 adult animal imaged at 60x. D) Embryonic cells in culture isolated from GN883 animals. E) GN884 adult animal imaged at 60x. F) Isolated cells from GN884. G) Brightfeild and wSc fluorescence overlay of *mec-17p*::puroR::sl2::wSc expressing animal imaged at 60x. H) Two dimensional histograms (FL1H versus FL3H) from debris gated flow cytometry data analyzed with classifer for GN883, N2, GN884, and OH441 animals. Green, black, and blue lines mark GFP(+) and live/dead cutoffs as described earlier.

GFP(+), these events are indistinguishable from those of the N2 sample and as such are likely dead, auto-fluorescent cells.

2.3 Discussion

Measuring changes in fluorescence expression or intensity across a heterogeneous population of cells is a powerful approach for genetically profiling cellular subtypes. The nematode *C. elegans* is an excellent model system for conducting such experiments however, conducting these studies with

cells dissociated from embryos remains an underutilized experimental approach. This is likely due to the cell culture system having a reputation for being technically challenging or requiring significant expertise. Additionally, tools for studying isolated cells, such as FACS, have primarily been optimized for mammalian cells with little information available describing how to accommodate the needs of nematode cells. Despite their size and autofluorescence, the results presented here suggested that nematode cells can be automatically classified into subgroups with custom code utilizing open source flow cytometry analysis libraries. Having established that this computational system could accurately classify the response to a cytotoxic challenge (puromycin, heat, ethanol), I further asked if it could resolve smaller changes such as sensitivity to puromycin treatment. I found measures of population enrichment and survival were subject to substantial variability at lower levels of puromycin treatment. This may suggest that the system is best suited for binary assays where there is a dramatic difference between conditions. However, this resolution limit may be based on other factors, as is described in the discussion to follow. Regardless, the work presented here establishes an easily adopted, high-content and -throughput, reduced-biased assay for measuring population wide fluorescence in isolated *C. elegans* embryonic cells. As the code is freely available for download, the techniques can be adopted for glycobiology, neurotoxicity, and genetic profiling experiments with existing worm strains, thus expanding the *C. elegans* researcher's toolbox.

2.3.1 Limitations

While testing different protocols for treating cells with puromycin, I observed a direct trend between culture conditions and GFP expression. In early experiments, I maintained cells in culture for up to three days prior to analysis. In each of these cases, I was unable to observe a distinct GFP(+) cluster as I had in the experiment presented in Figure 2.8. Similarly, when GFP(+) cells were treated with either intense heat or high levels of puromycin, they were also significantly dimmer. This is in opposition to the trend observed in wild type cells where auto-fluorescent cells appear brighter when dead. These changes in fluorescence intensity with cell viability make it extremely difficult to confidently classify dead cells as GFP(+/-). Such a shift in overall brightness is easily seen in Figure 2.7.D where even after auto-fluorescent gating the puromycin treated OH441 sample is skewed to

the left of the control sample. Here, the GFP cutoff overestimates the cutoff for the true distribution of the GFP(+) population whereas this line is perfectly positioned for control cells. Thus, it is entirely possible to detect dead cells, however it is not so easy to quantify what percent of the dead cells are GFP(+). This poses a challenge if your goal is to measure relative changes in survival rates.

An additional point for consideration is that this algorithm does not use spillover compensation in a conventional sense. Although the classifier does have code for calculating a spillover matrix and was tested for such purposes (see Appendix A), it is not actively used for identifying cells. In this work, compensation produced unreliable events likely due to the high levels of autofluorescence as suggested in [99, 101]. The presence of this autofluorescence population results in a mathematically nonsensical spillover matrix that when applied to flowSets results in numerous events becoming negative in value and thus unable to be analyzed. Although the lack of compensation did not pose a serious issue for the purposes of this work, thanks to the auto-fluorescent gating strategy developed, other samples may require compensation for accurate classification. This would likely be the case if expanding analysis to more than two fluorescent channels, which would pose even more challenges since the complexity of analysis scales at least exponentially rather than linearly. For each fluorescent channel used, a "fluorescence minus one" control must be developed and used as a training set. For example, adding measurements for the orange (FL2) channel would require producing orange and green, orange and red, and just orange data sets. Therefore extending this existing algorithm to analyzing more than two challenges would require further investment in handling experiment complexity.

2.3.2 Future Directions

The work presented in this chapter provides a foundation for numerous techniques not previously established for this system. In that regard, there are substantial opportunities both for improvement and for extension to additional applications. Early in the chapter, I showed the versatility of this system for live cell imaging and cryoET. Figure 2.1 panel shows the successful identification of mitochondria within the isolated TRNs, suggesting it might be well suited for studying mitochondrial transport and activity in real time. The results of our preliminary cryoET studies indicate

that these very small cells are perfectly suited for this application as the cells can adhere to standard TEM grids and are slim enough to be imaged without FIB milling. Others have previously demonstrated how AFM can be used to measure mechanical properties of the isolated cells [24]. I have built on this knowledge by demonstrating pairing AFM measurements with calcium imaging and measuring surface topology of living cells. Combined with previously established knowledge, this system is clearly well suited for measuring binding interactions of surface proteins, and potentially the resulting depolarization of the cell membrane in response to those binding events, using functionalized AFM probes. Applications aside, there is substantial room to improve cell culture conditions, such as media pH as this is known to have dramatic effects on cell health [104]. Similarly, to my knowledge no group has published efforts to improve neurite outgrowth or cellular development with the addition of growth factors to media. These are common approaches when working with mammalian neurons and may be similarly important for this system.

There are several potential avenues for improving upon the developed classification algorithm. From the completed experiments, I found that it was critical for training data sets to be prepared in exactly the same manner as experimental samples. Slight variations in ArC dye preparation, for example, could significantly reduce the overall brightness of calibration beads (Figure 2.5 panel D). Optimizing the training data sets could therefore help to improve the accuracy of the classification process. Similarly, optimizing the process for labeling cells with ArC or PI may help improve classifier accuracy, as there are reports that incubation time with PI can increase the rate of false positives for dead cells (personal communication with FACS technicians). Adjusting the measurement process so that cells are first labeled with ArC and then immediately fixed prior to flow cytometry analysis, as described in [100] may help to overcome this challenge. In detecting dead cells, I also suggest that future efforts are placed into optimizing the size gating procedure to increase confidence in what events are analyzed as true cells. In developing the algorithm, I performed an exploratory sensitivity analysis to test how size gating parameters would affect the classification process. I found that lethally challenged samples had a greater proportion of events with lower FSC-H values (Appendix A), suggesting that these samples either had more debris or that dead cells are often smaller. Thus, this point in particular warrants further consideration. To overcome issues of detecting dead GFP(+) cells, staining cells with an antibody against GFP may help to amplify

the endogenous signal and thus improve detection. Finally, this algorithm does not employ common normalization strategies as described in [98] because I did not want to program an assumption of the populations present. However, these normalization techniques have benefits and future directions should consider ways of incorporating them into the classification process.

The initial goal in developing the classifier presented here was to measure changes in population wide GFP(+) expression following puromycin treatment. We hypothesized the puromycin selection could be used to enrich the isolated *C. elegans* embryonic cultures for cells of interest by expressing the puroR gene under cell specific promoters. We tested this hypothesis using animals expressing the puroR gene under the TRN specific *mec-18* promoter and saw promising results, however the fluorescent marker in these animals was expressed under a distinct TRN promoter, *mec-17p*. Thus it is not possible to directly correlate GFP(+) expression with puromycin selection. To circumvent this, I developed strains expressing both puroR and reporter genes under the same promoter. However, the promoters chosen in two of the cases did not have high enough expression levels at the measured stage to allow for detection with only a single copy of the transgene. Future directions could likely address this issue by using common *C. elegans* transgenic tools to introduce multiple copies of the transgene, thereby greatly increasing overall expression levels. Additionally, since this work was completed there have been substantial advances in the availability of single cell RNA sequencing data. These data may well reveal stronger promoters that could be used. Bioinformatics approaches could be used to mine this data to identify ideal promoters that have both cell-specific and sufficiently high enough expression levels. Based on the results shown in Figure 2.8 panel we see that for certain promoters expression levels are high enough to allow detection with only single copy insertion of the transgene.

2.4 Materials and Methods

2.4.1 Worm husbandry

Animals used for imaging were grown on standard NGM plates seeded with OP50 *E. coli*. Animals used for embryo dissociation were first chunked from dauer plates onto the NGM plates with OP50 until they reached a gravid state. At this point they were bleached and seeded on the same

type of plates. Once this second generation became gravid the animals were again bleached, but this time seeded on enriched peptone (EP) plates with NA22 *E. coli*. This step was then repeated to expand the total number of worms to roughly 25-50,000 before isolating the embryos for dissociation and cell culture. Table 2.1 lists the *C. elegans* strains used in this chapter.

Table 2.1: Worm Strains Used in Chapter 2

Strain Name	**Genotype**	**Purpose**	**Source**
GN595	pgSi71[*unc-119(+)pMec-18*::Puro::let-858] II; uIs31 III	Cloning of the puromycin resistance gene	Unpublished strain developed by M.Krieg in the Goodman Lab
GN692	ljSi1[*Pmec-7*::GCaMP6s *cb-unc-119(+)*] II; *lite-1*(ce314) X	Calcium imaging with BioAFM	[81]
GN883	pgSi131 [*vha-7p*::puroR::sl2::mNG] II	Visual confirmation of puroR genetic expression in hypodermal cells	This study
GN884	pgSi132 [*cmk1-p*::puroR::sl2::mNG] II	Visual confirmation of puroR genetic expression in neurons	This study
N2	Wild-type	Live/Dead and GFP(-) analysis	CGC
NM3573	jsIs1073[*pmec-7* TagRFP-mito *CBunc-119*]	Mitochondria analysis	[90]
OH441	*unc-119p::GFP*	Pan-neuronal GFP(+) analysis	[93]
TU2769	*mec-17(uIs31)* III	CryoET, TRN only GFP(+/-) analysis	[21]

2.4.2 Cell culture

Cells were prepared according to a protocol adapted from [42, 88, 89] with some modifications. All dissociation and cell culture steps were performed at room temperature using aseptic technique on a standard laboratory bench. Tightly-synchronized, gravid adult worms were rinsed from EP plates with distilled water and collected into 15 ml tubes. A dense pellet of worms was formed by spinning the tubes in a swinging bucket centrifuge were spun down at 350g for 3 minutes. The supernatant was aspirated off to the 3 ml line, leaving the worm pellet and some water behind. Bleach (1 ml) and 10 M NaOH (250 µl) is added to the tube, followed by enough distilled water

to bring the total volume to 5 ml. Next, the tube was capped, vortexed, and incubated for at least five minutes. At that time, the solution was closely monitored to find the soonest time that at least 80% of the worms were lysed. At that moment, 1X egg buffer was added to the tube to quench the lysis, bringing the final total volume to 15 ml. The resulting embryos were rinsed with a sterile 1X egg buffer three times prior to further isolation from lysis debris using a sucrose gradient. Following the sucrose gradient isolation, embryos were rinsed once more with sterilize-filtered Milli-Q water and transferred to microcentrifuge (1.5 mL) tube for the chitinase digestion step. In this step, embryos are suspended in a chitinase solution while slowly rotating for 50 minutes, at which point a small sample was taken to monitor egg shell digestion. When nearly all of the egg shells had been digested, the mixed population of isolated cells, exposed embryos, and larvae were rinsed with cell culture media and physically dissociated. To start, the solution was triturated with a 1000 µl tip, followed by multiple passes first through a 21 gauge and then 25 gauge needle. After a first round of dissociation, the solution was observed at 10x magnification to determine if additional trituration was necessary. Once physical dissociation was deemed complete, the solution was filtered through a membrane with 5 µm pores. This final cell suspension was spun down in a bench-top centrifuge at 900 g, 4°C, for 3 minutes. The supernatant was replaced with 500 µl of fresh cell culture media and the cell density checked using a hemocytometer at 10x magnification. The degree to which further dilution was needed to achieve appropriate cell density was empirically determined, as these small cell bodies (3 µm) do not allow for accurate cell density counts.

2.4.3 Cryo-electron tomography of isolated cells

Gold TEM grids with R2/2 Quantifoil Holey Carbon films (Electron Microscopy Sciences, Q225AR2) were prepared for cell culture by incubation in a solution (500 ug/ml in sterile, distilled water) of peanut agglutinin (PNA) for 30 minutes, followed by multiple rounds of rinsing with sterilized Milli-Q water. To seed cells, grids were placed in a sterile polystyrene dish and a small droplet of suspended cells were placed above on top of the grid and allowed to settle for a few minutes before filling the dish with additional cells in solution and media. Cells were allowed to grow for three days prior to plunge freezing. Given our findings that neurite outgrowth was essentially complete within 24 hours, plunge freezing could be performed sooner. Plunge freezing,

sample preparation, and imaging were carried out by our collaborator, C. Zhang, Skinitois Lab.

2.4.4 Combined atomic force microscopy and fluorescence imaging of live cells

For AFM experiments cells were prepared as described above, but seeded onto 60 mm, shallow glass-bottom Petri dishes (Willco). All experiments were conducted using Peak Force Nanomicroscopy Live Cell probes from Bruker (k= .08 N/m, 130 nm diameter tip, part no. PFQNM-LC-A-CAL) using a Bruker BioScope Resolve AFM (CSIF, Stanford) mounted on a Zeiss inverted fluorescence microscope. Calcium imaging was acquired by paired recordings between the devices. The user programs a script from within the Bruker software on the first computer to send a trigger pulse to the Zeiss software on the second computer. The user defines when this pulse is sent relative to the sequence of probe actions. I programmed this script to begin fluorescence recording just prior to the positioning the AFM probe to the specified location for stimulus application. Topographical scans of cells were acquired by scanning cells in non-contact mode. Images of scans were generated by analyzing *.spm files with the open source software Gwyddion.

2.4.5 Flow cytometry data acquisition

All flow cytometry samples were analyzed on BD Accuri C6 (Shriram Shared Equipment Room 068, Stanford University) at medium flow rate with default flow core settings. The system was primed for use prior to sample analysis by flushing with detergent and then Milli-Q water according to manufacturer recommendations.

2.4.6 Puromycin treatment

Puromycin was prepared from 10X stock by serial dilution into sterile cell culture media. Following the final dissociation steps, cells were divided into separate tubes for different concentration treatment and standard cell culture media was exchanged with puromycin-containing media of varying concentrations. Cells were incubated with the puromycin media in suspension for three hours with gentle rotation. Following incubation, cells were thoroughly rinsed and resuspended in standard media, and seeded onto surfaces treated with poly-l-lysine as suggested by [88] for FACS

experiments. Just prior to flow cytometry analysis, samples were dislodged from cell culture substrates using gentle trituration and then collected for labeling.

2.4.7 Live/dead labeling

For propidium iodide (PI) labeled cells, 5 ul of PI (1 mg/ml) was added to each sample that contained cells resuspended in 1 ml of PBS. For ArC stained cells, samples were allowed to incubate with 5 µl of ArC, prepared according to manufacturer specifications, for 30 minutes before cells were rinsed and media replaced.

2.4.8 Computing environment

Code was developed in R Studio and is presented in Appendix A. Final figures were generated in macOS Catalina, version 10.15.7 with a 2.3 GHz 8-core Intel Core i9 processor and 16 GB 2667 MHz DDR4 memory. Final analysis code was ran with BiocVersion 3.12.0.

2.4.9 Molecular biology and worm strain development

Transgenic animals were developed using a protocol and flexible targetting vector develop by A. Das, based on the methods described in [103]. Each plasmid was designed in ApE using promoter and gene sequences from WormBase.org (release WS268). Primers used for cloning genes were designed by JF and synthesized by the Protein and Nucleic Facility (Stanford University). All plasmids were generated using the same workflow. Primer sequences, recommended annealing temperatures, and details of the restriction enzymes used for synthesis are detailed in Table 2.2 below. Details of the plasmids generated using these methods, with names assigned by MBG, are detailed in Table 2.3. All primers, plasmids, and worm strains described are available in a repository maintained by the Goodman Lab and are available upon request.

General procedures for worm strain generation were as follows. Genetic sequences were amplified either from lysed wild-type or transgenic animals, or from existing plasmids using the primers in Table 2.2 using KOD Hot Start buffer and Proteinase K. DNA was PCR amplified (10x KOD Hot Start kit, MilliPore Sigma), analyzed by gel electrophoresis, and purified from gel using a ZR

Plasmid Miniprep. Purified DNA products were assembled into a single construct using the Gibson Assembly Protocol (NEB, E5510) as described in [105]. Following assembly, competent cells (DH5-alpha, NEB) were transfected with the assembled vector and then seeded onto agar plates with ampicillin, and grown overnight at 37°C. The next day, single colonies were picked from these plates and transferred to LB media for colony expansion. These colonies were grown overnight at 37°C with shaking. The next day, plasmids were purified from their respective colonies, the plasmid size was verified by gel analysis, and plasmids were sent out for sequence verification. Once verified, plasmids were injected into young adults using standard microinjection techniques and allowed to grow on NGM plates for three days. At this time point, hygromycin was added to the plates to eliminate F1 animals lacking the desired construct that contains a hygromycin resistance gene. On the fourth day following hygromycin exposure, surviving animals were singled and transferred to new NGM plates. Animals were screened visually to verify expression of fluorescent reporters. Animals were lysed and the resulting products were transgene insertion was verified by PCR. Once verified, animals were heat shocked to remove the self-excising cassette.

Table 2.2: Primers used for gene cloning and plasmid synthesis

Primer	Length	Tm	Sequence Details	Sequence
JF001	56	58	*(mec-17p)*-nheI-puroR-f	ccgatggatcagttgtgagacgattcgatcgctagc ATGACCGAGTACAAGCCCAC
JF003	57	58	*(cmk-1p)*-nheI-puroR-f	CTTACTCAACGGCTCCCCCTCGACGA GCCCAgctagc ATGACCGAGTACAAGCCCAC
JF004	55	62	sl2-afIII-puroR-r	gttaactaggtgaaagtaggatgagacagcctact taagAGGCACCGGGCTTGCG
JF005	24	57	stop-sl2-f	taggctgtctcatcctactttcac
JF006	51	58	mNeonGreen-sl2-r	CATGTTGTCCTCCTCTCCCTTGGAGA CCATgatgcgttgaagcagtttccc
JF007	58	68	(backbone 465)-sphI-*cmk-1p*-f	cgtcgatcatcctgtaaaacgacgg ccagtgcatgcgctggtgggaagatgagtgatg
JF008	22	71	nhe1-*cmk-1p*-r	gctagcTGGGCTCGTCGAGGGG
JF011	20	70	puro-f	GCGCAGCAACAGATGGAAGG
JF012	21	59	puro-r	GAGTTCTTGCAGCTCGGTGAC
JF013	61	61	(backbone 465)-sphI-*vha-7p*-f	CGTCGATCATCCtgtaaaacgacggccagtgcat gcAGGAAATTGTGAGAAGA-GAAATTTg
JF014	56	62	puroR-*vha-7p*-r	GAGGCGCaCCGTGGGCTTGTACTCGG TCATgctagcATTACGTCGTTGGTG-GAttg
JF015	21	67	(-227 bp)-*vha-7p*-f	CGCCTACAGTGCTCAAATGC
JF016	52	71	puroR-*cmk-1p*-r	GAGGCGCaCCGTGGGCTTGTACTCGG TCATgctagcTGGGCTCGTCGAGGGG
JF017	46	73	(171 bp)-*vha-7p*-r	gaacgttcagactgttggcttgttt cctgaatttttgaaaaacttc
JF018	26	65	(58 bp into *vha-7p*)-r	ccgagcccaaaagtacttaaatatcc
JF019	56	64	cyOFP-avrII-sl2-r	CTTAATAAGCTCTTCTCCTTTTGACAC CATcctagggatgcgttgaagcagtttcc
JF020	55	62	ttTi4348-Sph1-*mec-17p*-f	CTATAGGGCGAATTGGGCCCTCTAGAT gcatgcGTTTTCAATACCGTCCAACATG
JF021	51	63	puroR-AvrII-*mec-17p*-r	CACCGTGGGCTTGTACTCGGTCAT cctaggGATCGAATCGTCTCACAACTG
JF022	51	67	*mec-17p*-AvrII-puroR-f	GGATCAGTTGTGAGACGATTCGATC cctaggATGACCGAGTACAAGCCCAC
JF023	50	79	sl2-puroR-r	GTTAACTAGGTGAAAGTAGGATGAGA CAGCTCAGGCACCGGGCTTGCGGG
JF024	52	62	puroR-stop-sl2-f	CTGGTGCATGACCCGCAAGCCCGGTG CCTGAGCTGTCTCATCCTACTTTCAC
JF025	53	64	wSc-ClaI-start-sl2-r	GATAACTGCCTCTCCCTTGCTGAC atcgatCATGAT-GCGTTGAAGCAGTTTCC
JF026	46	69	sl2-puroR-r	GTTAACTAGGTGAAAGTAGGATGAGA CAGCTCAGGCACCGGGCTTG

Table 2.3: Plasmids developed for cell specific expression of puroR gene and fluorescent reporter

Plasmid name	**Genotype**	**MosSci Site**
mg662	*mec-17p*::puroR::sl2::mNeonGreen	ttTi5605
mg667	*vha-7p*::puroR::sl2::mNeonGreen	ttTi5605
mg668	*cmk-1p*::puroR::sl2::mNeonGreen	ttTi5605
mg683	*mec-17p*::puroR::sl2::cytoOrangeFluorescentProtein	ttTi5605
mg699	*mec-17p*::puroR::sl2::wormScarlet	ttTi4348

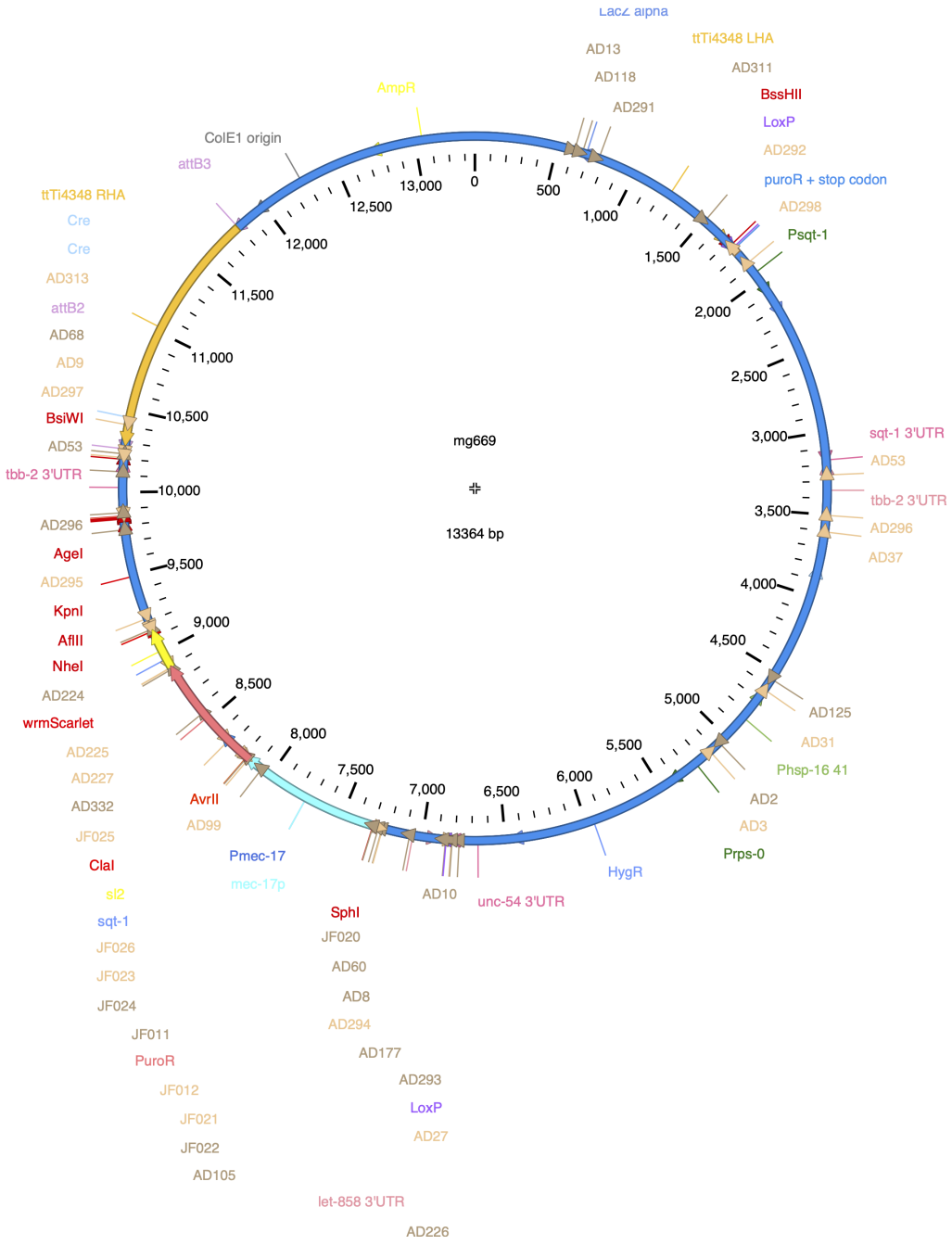


Figure 2.10: **Flexible vector for CRISPR/Cas9 mediated single copy insertion of cell-specific puroR transgene and fluorescent reporter**.

Chapter 3

Development of a micropatterning protocol for reducing TRN morphological variance

3.1 Introduction

Mechanoreceptors (MR) are a class of cells, most often neurons, responsible for detecting and communicating information about physical stimuli from their environments. Though biological in their origins, these MRs share many properties with engineered sensors and are easily classified into subgroups based on their electrical signatures. It has long been established that each class of MR has specific characteristics such as sensitivity, dynamic range, and frequency selectivity. However, little is known about the principles that enable biological MRs to achieve functional specificity. From *in vivo* and *in silico* experiments we hypothesize that specialization is directly related to not only the position of mechanosensing molecules, but also the overall morphology of the MR. Wanting to overcome the practical limitations of investigating a potential role for MR shape *in vivo*, we aimed to develop a platform for controlling *C. elegans* TRN process outgrowth *in vitro*. Progress in this area is limited by a lack of tools for controlling the morphology of MRs *in vivo*, and by a limited understanding recapitulating the distribution of mechanosensory molecules *in vitro*. Micropatterned

substrates have often been employed in experiments using mammalian cells to control cell shape, but no group had previously tested if this method could be applied to cells isolated from *C. elegans*. The theoretical existence of form-function coupling remains an interesting and unanswered question in the field of mechanosensation. Testing for such a relationship will not only help to build our understanding of how the mechanosensory process works *in vivo*, it can provide a template for engineering biological sensors.

3.1.1 Background

At its most basic level, mechanotransduction can be defined as the conversion of physical stimulus such as a push or pull into neural impulse. Though it is easiest to consider this sensory process as it relates to our conscious experience of touch and sound, it occurs continuously without our recognition all throughout the human body. For example, we are not consciously aware of the mechanosensation associated with innervation of internal organs such as the stomach that allows the central nervous system to detect and respond to changes in overall size. Though these free nerve endings share a common function, they show high degrees of functional specificity. Even with a single type of sensory cell, for example hair cells of the inner ear are tuned to specific sound frequencies. This means that the characteristics of their electrical signals as sensors will vary based on the waveform of the applied stimuli. One example of this is with TRNs where their mechano-electrical transduction channels are optimized to function as vibration detectors, but only for specific frequencies. Interestingly, these electrical signatures are conserved between species, suggesting a type of conserved MR function. The TRNs in *C. elegans*, for instance, respond to the step-inputs of constant force with the same onset and offset characteristics of mammalian Pacinian corpuscles. This idea of conserved MR function is further supported by the fact that different species have bristles or whiskers that are physically coupled to sensory afferents. Another shared property of MRs across phyla is that they take on unique architectural characteristics that often compliment their surrounding tissues. Drawing from engineering sensor design principles we hypothesize that MR shape may in part contribute to the responsiveness of the mechanosensor to physical stimuli.

Determining the role of MR morphology on mechanosensation hinges on the ability to experimentally alter shape and then measure functional changes of the MR. Cell culture is ideally suited

as a platform for pursuing this inquiry, as it affords researchers a greater level of control over the developmental environment. The nematode *C. elegans* is one of the few where specific genes have been identified which, when mutated do lead to changes in MR morphology. In the case of TRNs, the extracellular matrix (ECM) protein MEC-5 is required for correct positioning of the mechano-electrical transduction (MeT) channels [28]. This protein not made by the TRNs but is secreted other cells, and thus if we alter the positioning of the TRN we inevitably alter the cell-ECM signaling that is necessary for MeT channel distribution. For *in vivo* experiments it is nearly impossible to decouple the effects due to changes in MR shape or morphology from those due to changes in local signaling. Conversely, there are a tools available that have been used to control the shape of a diverse cell types from different species. Many groups have developed methods for controlling the deposition of specific cell adhesion molecules (CAMs) in the cell culture environment such as microcontact printing and lift-off patterning [106, 107]. The result of these biofabrication processes are substrates that have been functionalized with CAMs in geometrically defined regions, thus constraining where cells are able to attach and what shape they acquire. We hypothesized that we could use these technologies to create a cell culture environment that allows us to both control cell shape and control the positioning of ECM proteins thought to play key roles in MeT channel localization. As there are currently no published studies that have used micropatterned CAMs to control *C. elegans* neurite outgrowth, we were motivated to investigate the feasibility of this experimental technique for studying form-function coupling in TRNs.

3.1.2 Summary

An an entry point, I translated experiment system requirements into design specifications. In this chapter I present the details of that protocol, including the design specifications and engineering methodologies employed during development. I further demonstrate that the minimum achievable feature size that is critical for working with nematode cells, especially neurons. The endpoint of this work is a procedure for generating micropatterns that constrain isolated and cultured TRNs to thin stripes. This reproduces their *in vivo* like unipolar morphology and straight neurites. The approach also suppresses secondary branching often observed when cells are grown on uniform surfaces *in vitro*. This restriction of TRN morphology further improves experiment robustness by reducing the

variance typically associated with cultured TRNs and matches digital image acquisition to use automated image analysis methods. Culturing TRNs on patterns rather than uniform substrates also improves experiment rigor the task of tracing neurites emanating from cell bodies is trivial when neurites are constrained to think stripes and do not intersect one another. Furthermore, I demonstrate the use of this approach for generating micropatterns with multiple CAMs, which provides researchers with a way to test if the presence of secondary CAMs is sufficient to recapitulate the *in vivo*-like distribution of mechanosensory molecules. The system developed here establishes a novel tool for expanding our understanding of the cellular and molecular basis of touch sensation.

3.2 Results

3.2.1 Design goals and system specifications

To develop a cell culture system enabling the analysis of MEC-4 distribution with minimal measurement noise, I took an engineering design approach by turning our experimental goal of determining if the subcellular distribution of MEC-4 observed *in vivo* was preserved *in vitro* into design specifications. First, the developed system needs to allow for the rapid identification of unique TRNs (to avoid repeat measurements) with similar morphologies. The system would ideally result in many individual neurons studied in each biological replicate. Since the TRNs occur at such a low frequency in cultures (less than 1% of the population [42]), increasing the desired number of cells per replicate means having a larger micropatterned area. Larger micropatterned increase fabrication and imaging times, while also increasing supply costs. Thus an important design rule is to balance these competing factors and developing a system that generates large numbers of individual samples (i.e., high-content), but minimized the time required for micropattern fabrication and cell imaging time. The system would also need to allow for: two-CAM patterning, minimal batch-to-batch variation, small feature sizes to control the unusually small neurites, and production of multiple CAM conditions in one fabrication run. This last design goal stems from the need to test the same biological replicate (i.e., a single dissociation) in more than one condition (e.g., testing for an effect of a second CAM and comparing this to a control condition).

The resulting process (Figure 3.1) is centered around the use of a commercial, Light-Induced

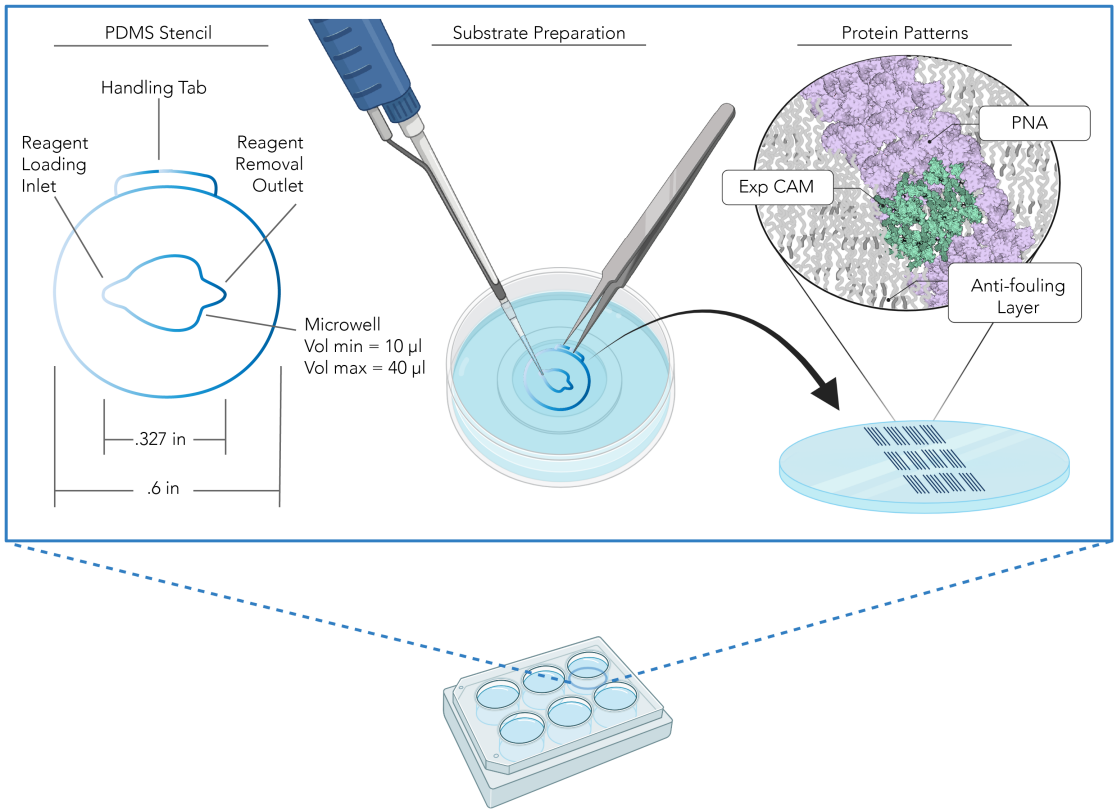


Figure 3.1: **High-content micropatterned cell culture system**. Micropatterns of cell adhesion molecules (CAMs) generated within a glass-bottom, six-well tissue culture plate that allows for a rapid fabrication-to-imaging workflow. PDMS stencils restrict the patterning area to minimize production costs while maximizing content and throughput. The micropatterns are oriented to minimize imaging time and reduce the risk of duplicate imaging. The system supports the addition of an experimental CAM with the PNA that is necessary for cell adhesion.

Molecular Adsorption system (LIMAP; PRIMO protein micropatterning, the Alvéole) to produce geometrically defined regions of proteins or CAMs on a user's substrate of choice [108]. For the substrate, I chose to make patterns on glass coverslips that come embedded within a six-well tissue culture plate. The total surface area of each well is too large for practical purposes in terms of production costs, time, and imaging, and so I designed a custom PDMS stencil to restrict the patterning area. The stencil has a handling tab for ease of application onto glass and a microwell for holding reagents and allowing access to the substrate for patterning. The microwell is designed

to hold between 10 and 40 µl of liquid, where 10 µl is the minimum volume needed to keep the surface fully hydrated. Two ports at each edge of the microwell provide directional flow of reagents as applied, and stability for the pipette tip during loading and removal. The PDMS area outside of the microwell was maximized to ensure adequate adhesion to the glass throughout the fabrication process. The patterns themselves were designed to encourage an efficient imaging workflow. The main feature of the patterns are long straight stripes that are just wide enough to allow the cell body to adhere to the pattern, but narrow enough to discourage wandering of the neurite across the line. Neurons are able to extend processes that span a gap without adhesion, and each line was spaced far enough apart from its neighbor to discourage the line-hopping observed in preliminary experiments.

3.2.2 Dual-CAM micropatterning protocol

The basic premise of all micropatterning techniques is to create a region of CAMs, to which cells can attach and spread, that is distinct from other areas of the substrate. These other areas are sometimes free of any coating, as is the case for microcontact printing where CAMs are "stamped" onto glass, or they are covered in an anti-fouling layer that passively blocks cell attachment. In many cases, the anti-fouling layer is first generated as a uniform coating across the surface that will be patterned. Then, select regions of the coating are degraded. In the case of PRIMO-based LIMAP, this antifouling layer is degraded by a photoinitiator (PLPP), activated by UV light. The reaction causes degradation of the antifouling layer when supplied with sufficient energy. When the unactivated PLPP is in place, the antifouling layer will be degraded wherever it is exposed to UV light. If too much energy is supplied the reaction can extend beyond the intended patterning area. PRIMO uses a digital mask (an example is shown in Figure 3.2) to guide the projection of UV light onto the desired substrate. Following exposure with UV light, the polymer layer that provides antifouling is cleaved and can be rinsed away. The exposed area can then be "back-filled" with a CAM of choice, so long as that CAM is able to adsorb to the substrate. The rough overview of this process, as it applies to the protocol developed in this thesis, is presented in Figure 3.2.

An important experiment goal motivating the development of this system was the desire to test for the role of a second molecule (in addition to one required for cell adhesion). Achieving this goal with LIMAP is relatively straightforward when using the anti-fouling layer and photoinitiator

previously described, however it has yet to be achieved with the more recently available methods that improve the LIMAP micropatterning procedure. The first of these is an alteration to the anti-fouling layer that prevents cells from attaching to the non-patterned regions of the substrate. The initial technique used by many in the field was to apply PLL-g-PEG in solution to a plasma treated surface in one step. The result was a brush-polymer layer of poly-ethylene-glycol (PEG) grafted to poly-l-lysine (PLL). This layer had little mechanical stability and quickly degraded within a few days. Alvéole recently provided resources for users to generate a more stable and long lasting anti-fouling layer by first depositing only PLL and then adding mPEG-SVA in a HEPES-buffered solution. Unlike the PLL-g-PEG anti-fouling layer, this new approach results in a layer that doesn't require constant hydration before patterning. The second advancement in micropatterning, from Alvéole, is a photoinitiator gel (PLPP gel) that reduces the required activation energy, and thus exposure time, by over 20 fold (decreased from 1000 mJ/mm^2 to 50 mJ/mm^2). The advantages provided by these two strategies motivated us to consider if we could develop a protocol based on the new approaches. The resulting protocol is presented as a process flow in Figure 3.2.

The process begins with plasma treating the glass-bottom tissue culture plate, with the lid off, to clean the glass and render the surface hydrophilic. The PDMS stencil and PLL solution must then be applied to the glass within ten minutes to take advantage of the surface activation provided by the plasma treatment. Following a one-hour incubation of PLL, the microwell must be vigorously rinsed with sterile distilled water (diH20) and then with pH-adjusted HEPES buffer. Once the HEPES is in place, the mPEG-SVA HEPES solution can be prepared and must be applied to the microwell within ten minutes due to the short half-life of SVA. After allowing this solution to incubate for one hour, the microwell is thoroughly rinsed with sterile ddH20 and allowed to air dry without exposure to light. Once dry, PLPP gel is applied directly to the surface of the antifouling layer and pure ethanol is added to the gel to help with dispersion. The microwell is again left to dry and can be stored in cool, dark conditions for at least 48 hours before patterning, although longer storage durations may be possible. Once the microwell is fully dried, patterning can be completed using energy dosages of 30-50 mJ/mm^2. Following light exposure, the unactivated PLPP gel must be rinsed away from the microwell using ddH20. Next, the antifouling layer needs to be rehydrated with PBS for at least five minutes prior to depositing the first CAM. Once CAM #1 has been applied and allowed

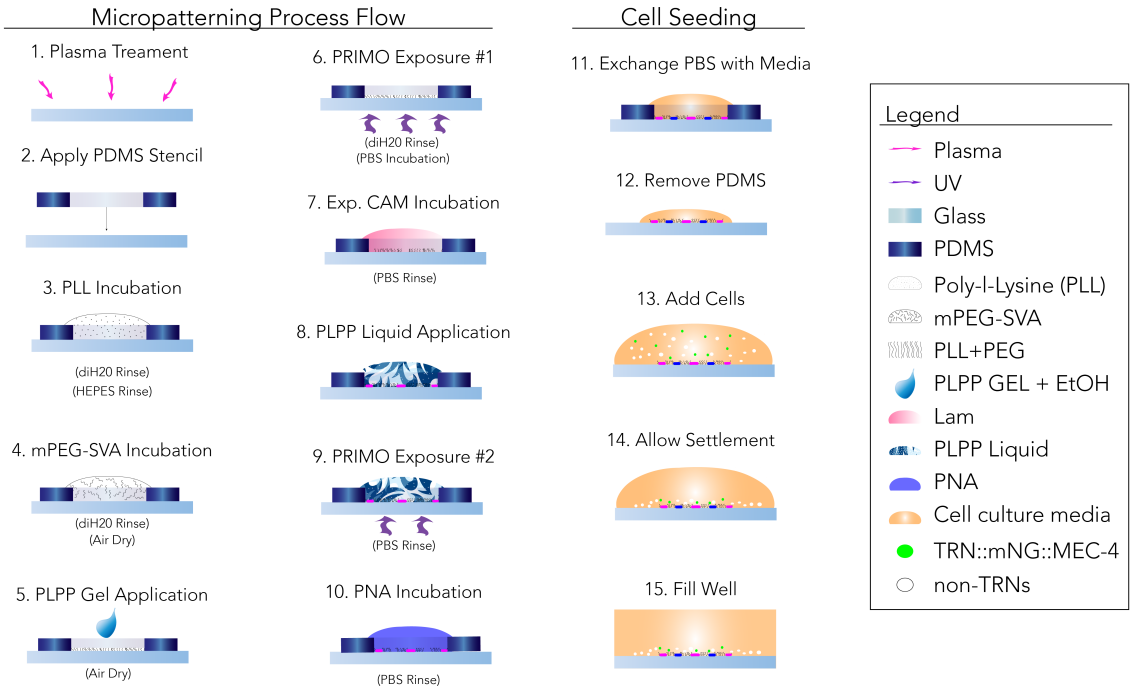


Figure 3.2: **Dual-CAM micropatterning process flow**. Glass substrates are prepared as described to generate micropatterns with two cell adhesion molecules. The fabrication process builds those described elsewhere [108, 109] and combines the two types of PLPP to optimize fabrication times.

to incubate (20 minutes), and the excess rinsed away with PBS, PLPP liquid can be applied for the second exposure. Since this exposure uses PLPP liquid it requires the standard UV dosage of 1000 mJ/mm^2. Once complete, the PLPP liquid is removed by washing with PBS and CAM #2 is applied and allowed to incubate (20 minutes). Once CAMs have been applied to the microwell, the antifouling layer does require constant hydration. The patterned substrates should be used as soon as possible as the antifouling layer will degrade over time. The precise lifetime of these patterns has not been characterized. Exact cell seeding techniques to ensure that cells are deposited over the patterned area will vary depending on density and should be optimized for specific applications. An suggested workflow as used in this work is shown in Figure 3.2.

3.2.3 Protein micropatterning has limited spatial resolution

The initial goal of our micropatterning cell culture experiments was to test for MEC-4 localization on patterns that mimic the spatial characteristics of TRN morphology and MEC-4 distribution *in vivo*. TRNs extend long, straight processes in the whole animal ($\tilde{5}$00 µm in length) that are extremely narrow in diameter (approximately 200 nm with some variation with organelle organization). The well quantified MEC-4 puncta are approximately 1 µm wide, spaced 1.5-4 µm apart (center-to-center) on average as visualized by fluorescence microscopy [61]. The morphological characteristics of the TRN neurites are at least an order of magnitude smaller than what has been observed in mammals and are far below the smallest resolvable feature sizes achieved with other micropatterning approaches [107]. The cell body of the attached TRN *in vitro* is typically on the order of 5 µm by my measurements and so I used this as an upper bound for the width of the lines. To test if the PRIMO system could achieve the 1-5 µm feature size needed I designed an exposure test pattern based on standard photolithography techniques. The test pattern (Figure 3.3) has progressively increasing feature sizes spaced apart with increasing distance. This allows me to determine not only the minimum feature size, but also how closely these small features can be placed for full resolution. Figure F:ch3.BM panel A shows the original pattern alongside the final protein micropattern that was made with fluorescently-labeled PNA (PNA-AlexaFluor594) using standard patterning procedures. Using the open source software ImageJ [110], I measured variations in fluorescence intensity across the patterned region with the Plot Profile tool (Figure 3.3, panels B through F). Through this analysis, I observed that the smallest features (3 pixel wide patterns) would need to be spaced at least 2.5 µm apart in order to be clearly resolved (Figure 3.3). Although the predicted width of these features is 1 µm based on the standard conversion factor (0.27 µm/pixel), in practice they are closer to 2 µm in width, but this exact value can vary greatly even within a single pattern.

3.2.4 PNA Micropatterns guide TRN morphology

The final process would need to meet two criteria: (1) TRNs adhere to patterned PNA and (2) Patterns are sufficient to guide TRN morphology. To test if these conditions were met with the

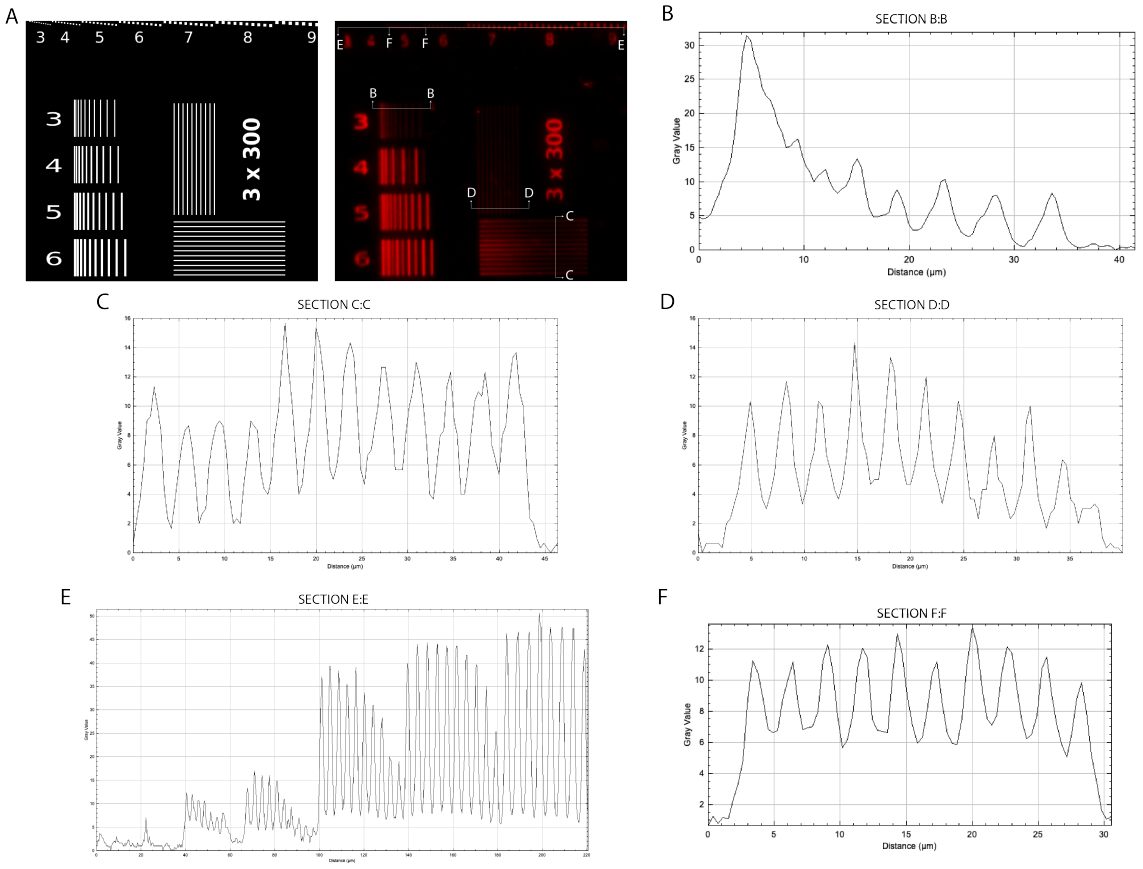


Figure 3.3: **Analysis of micropatterning resolution limits**. A) Detailed view of the custom test exposure mask (left panel) and the resulting protein micropattern (right panel). Micropattern image is annotated with line segments used to generate fluorescence intensity plots in B through F. B) Intensity profile for section B:B, showing peaks corresponding to 3 pixel wide vertical lines with increasing spacing. C) Intensity profile for section C:C, showing peaks corresponding to 3 pixel wide horizontally oriented lines, evenly spaced 3 pixels apart. D) Intensity profile for section D:D showing peaks corresponding to 3 pixel wide vertical lines, evenly spaced 3 pixels apart. E) Intensity profile for section E:E showing peaks for square patterns 5 pixels wide and larger. F) Intensity profile for section F:F, where section line is oriented along the diagonal of the pattern, showing peaks for 5 pixel wide squares evenly spaced 5 pixels apart.

designed patterns, I seeded cells on patterns fabricated using the process flow described in Figure 3.2 and specifications presented in Figure 3.1. In these experiments, I generated two types of patterns. The first used only fluorescently labeled PNA since this is the only CAM known to support the adhesion of cells from *C. elegans*. The developed process flow is easily adapted for

single CAM patterning, as it simply removes steps #7 through #9 in Figure 3.1. These patterns are made with PNA conjugated to a red fluorophore but are pseudo-colored in blue for clarity and viewing ease. The second set of patterns used a second CAM: laminin sourced from murine tumors (Engelbreth-Holm-Swarm). This protein is commonly used in mammalian cell culture [111] and is readily available in fluorophore conjugated form. The version used in these experiments is tagged with rhodamine (a red dye), pseudo-colored in magenta through this text for ease of discrimination. For these dual-CAM patterns, due to a limitation in the number of available fluorescent channels and fluorophore conjugated dyes, non-fluorescent PNA was used for the lines that guide neurite morphology. In Figure 3.4 the non-fluorescent lines of PNA are oriented horizontally, as suggested by the orientation of the neurites in panel A, upper right corner. Cells were prepared using the methods described in Chapter 2 from embryos expressing mNeonGreen tagged MEC-4 (mNG::MEC-4, GN753 pgSi116[mec-17p::mNG:MEC-4] II) [61]. Cells were cultured for 24 hours prior to imaging.

From qualitatively analyzing the cells in culture on patterns during imaging, I observed a strong correlation between neurite morphology and pattern geometry. The types of cells observed were the same for both patterning conditions in these experiments. Many of the TRNs extend neurites along the direction of the patterns, most usually in the center of the stripes, as seen in Figure 3.4. The line widths are sufficiently large enough to allow the cell body to attach to the substrate and extend up to three processes. The majority of TRNs in culture are unipolar, with morphological variation matching what has previously been reported [47][83]. There are instances of bipolar and psuedo-unipolar cells for all conditions tested at roughly the frequencies expected based on published findings. Consistent with these reports the cells mature rapidly, reaching a length of 50 μm, on average within 24 hours of seeding. A minority of the mNG::MEC-4 expressing cells had only truncated neurites, <15 μm in length. There was sufficient mNG::MEC-4 fluorescence distributed within neurites to visualize growth cones and lamellipodia. However, these membranous extensions were not present for all TRNs. Although the neurites generally followed the PNA stripes, there are instances where they veer into what should be the antifouling layer at the distal tip of the process. An example of this is shown in Figure 3.4 (the neurite tip bends down towards the scale bar). In some cases the TRNs contacted other cells in culture, either via the soma or the neurite, shown in the brightfield images

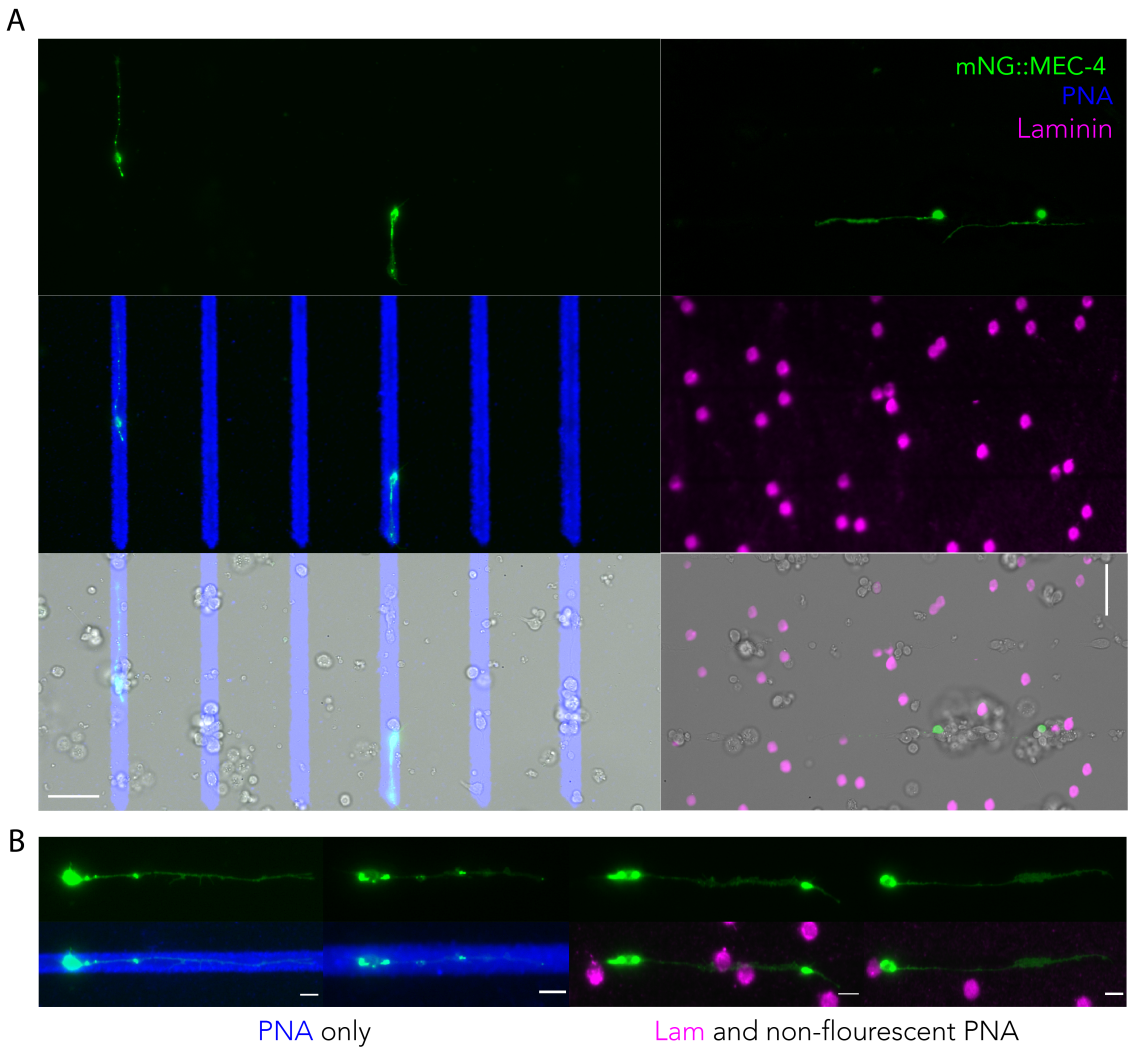


Figure 3.4: **PNA Micropatterns guide TRN morphology.** Representative results showing wide-field view mNG::MEC-4 expressing TRNs cultured on patterns. Cells in the left column are patterned on fluorescent PNA only. Cells in the right column are on non-fluorescent PNA patterned as horizontal lines, with fluorescent laminin dots. Bottom row overlay is the merge of green and red fluorescent channels with the brightfield image of the corresponding area. Scale bar is 20 µm. B. Digitally magnified views of individual cells on patterns cultured in the same conditions as described for panel A. Scale bars are 5 µm. Color coding of proteins is the same in all figures (green = mNG::MEC-4, blue = PNA, magenta = laminin.

in panel A of Figure 3.4. Future studies will be needed to clarify the mechanism and significance of these atypical events.

3.2.5 EHS Laminin micro-dots do not alter mNG::MEC-4 expression

Findings from previous studies suggested that three genes encoding extracellular proteins in *C. elegans* are necessary to position MEC-4 along the neurite [28]. Given that one of these genes encodes a collagen, we reasoned that other basement membrane proteins might be involved. Laminin is a highly conserved protein that is known to be present throughout the worm's body as part of the basement membrane throughout all of development [112]. Since mutations to the laminin encoding genes are typically either embryonic lethal or result in sterile adults, laminin mutations would not be detected in forward genetic screens for touch insensitive adults. To investigate a potential role for laminin, we asked if introducing laminin into the cell culture environment in the form of randomly distributed micro-dots could change the overall expression of mNG::MEC-4 observed along the neurite. To do this, I generated patterns and seeded cells according to the methods described above. Using the custom code described in Chapter 4 of this thesis, I quantified the lengths, average neurite intensity, and inter-punctum intervals for TRNs cultured on PNA stripes alone or in combination with laminin micro-dots. Neurite length was measured as the distance from the base of the neurite at the soma to the distal tip of the process. This region alone was used for quantifying mNG::MEC-4 expression via average neurite intensity and inter-punctum interval (IPI). Distributions for both measures of mNG::MEC-4 expression were indistinguishable between these two conditions, suggesting either that this laminin isoform or this pattern of laminin dots is not sufficient by itself to influence the distribution of mNG::MEC-4 channels in cultured TRNs.

3.2.6 Laminin-PNA grids increase TRN morphological variance *in vitro*

To test for this effect of the laminin dot pattern, we modified the design such that laminin was deposited in stripes orthogonal to PNA (Figure 3.6). Since earlier reports had found that the cells would not attach to laminin we chose to pattern the PNA second so that there would be no doubt about the bioavailability of the CAM. In attempting to image the cells to repeat the quantification

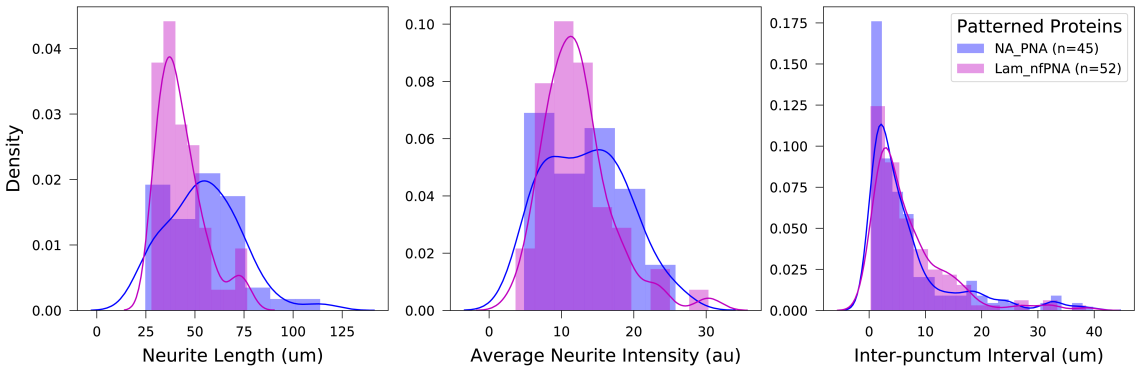


Figure 3.5: **EHS Laminin micro-dots do not alter mNG::MEC-4 expression** Quantification of morphological properties of TRNs cultured on either only PNA lines (NA_PNA) or PNA lines with randomly distributed micro-dots of EHS laminin (Lam_nfPNA). Kernel density estimates (solid line) are derived from underlying distributions (histograms) using Python seaborn.kdeplot method.

described above I found that there were very few TRNs that fit the experimental criteria. Unexpectedly, many of the TRNs formed attachments to the borders of the laminin lines, extending neurites orthogonal rather than parallel to the non-fluorescent PNA lines (Fig 3.6.A). In control experiments, the majority of TRNs do extend processes along the PNA lines and rarely show dramatic shifts in direction. This is the opposite of what was observed here where many of the neurites abruptly changed direction, acutely veering off path, or circling back on itself. The circling back behavior is extremely uncharacteristic. Despite what these specific examples may suggest, the cell culture population as a whole did not attach to the laminin lines, as shown in Figure 3.6. In viewing the patterned region at lower magnification it's clear that cells preferentially attach to the non-fluorescent PNA lines and not the horizontal laminin lines (magenta).

3.3 Discussion

For experiments in which biological variance increases the difficulty of deriving statistical certainty, experiment throughput is of great importance as this is a necessary means to generate large data sets. As described more thoroughly in Chapter 2 of this thesis, *C. elegans* cell cultures are inherently such a system as the total population of cells is a heterogeneous mix of cell types at

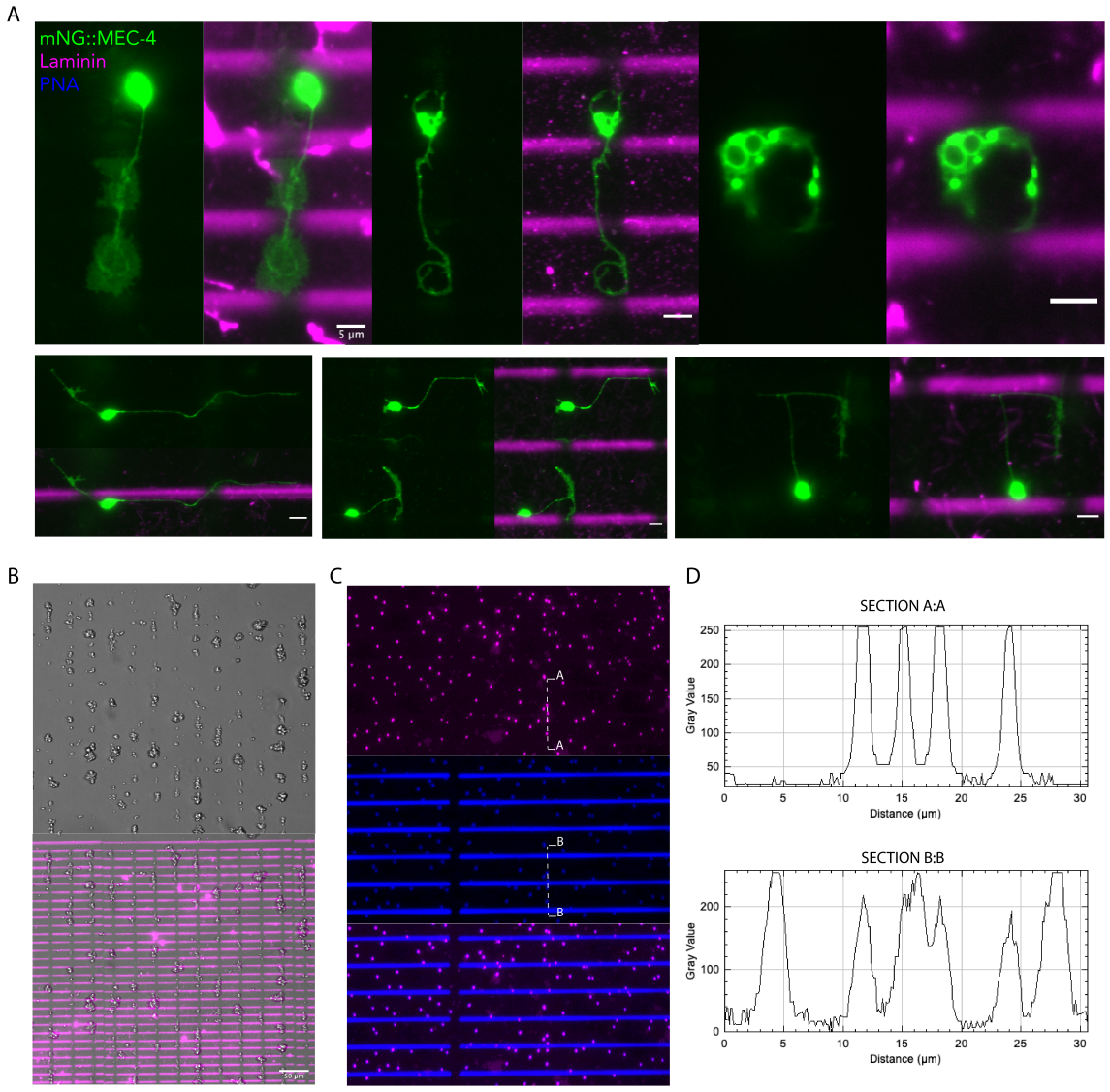


Figure 3.6: **Laminin-PNA grids increase TRN morphological variance *in vitro*.** A. Representative images of mNG::MEC-4 expressing TRNs on grid patterns. Non-fluorescent PNA is adsorbed to patterned regions running orthogonal to the horizontal, magenta lines (patterned laminin). TRNs show greater variance in the directionality of neurite outgrowth than in control conditions (i.e., only PNA). Scale bars are 5 µm. B. Zoomed out widefield view of micropatterned region shown in A. Cells preferentially attach to non-fluorescent PNA lines which are oriented vertically along the page. Scale bar is 50 µm

various developmental stages. Additionally, the small size of these cells increases the noise floor on morphological measurements since the width of TRN neurites in culture are between the theoretical diffraction limit and the practical limits of most microscopy systems. Generating large enough data sets for cells on micropatterns is directly related to the fabrication throughput of the chosen micropatterning process. Another design requirement was the necessity for an efficient and cost-effective fabrication process that could support high throughput assays. Since the target application for this system is to conduct large scale genetic screens, it was critical that a single production run be able to generate enough technical replicates to test two conditions in parallel. Even a robust fabrication process will have batch-to-batch variations that require the use of controls for each experiment until there is sufficient evidence to support pooling the data. With these design constraints in mind, I developed a micropatterning protocol and cell culture system for reducing TRN morphological variance, increasing experiment content, and measurement throughput.

Among established protein patterning techniques, LIMAP patterning is the fastest approach and provides the smallest resolvable feature size. Building the protocol around this technique substantially reduces the production overhead in terms of reagents, tool time, and the need for operator expertise. Using a glass-bottomed multiwell plates rather than individual coverslips provides an easy means for securing substrates for dual protein pattern registration–one of the most difficult details of producing multi-protein patterns. These multiwell plates use standardized sizing so that they can be loaded into universal microscope mounts that use springs to fix the plate position in the xy-plane. Hence, small taps to the plate during rinsing steps will minimally disrupt pattern alignment and overall batch-to-batch variation should be reduced. The glass bottoms also allow for a direct fabrication, to cell culture, to imaging workflow doesn't require transferring samples or complex mounting strategies. Since *C. elegans* cells do not require active temperature or C02 control for survival, the large wells that hold substantial amounts of media means there are fewer restrictions for imaging session duration. The PDMS stencil design reduces the reagents needed per production run and provides features to stabilize the pipette tips during fabrication. The chosen pattern layout speeds up the imaging process because it allows the operator to take a straight line path and keep track of position to avoid repeat imaging. The lines are narrow enough to control the morphology of the very thin neurites and are spaced far enough apart to minimize cross over between tracks and

close enough to enable efficient and systematic imaging.

3.3.1 Limitations

Cultured *C. elegans* cells are known to adhere to substrates treated with peanut agglutinin (PNA), a lectin commonly used for staining a wide variety of cell types. PNA binds specifically to glycosylated molecules possessing the Gal-β(1-3)-GalNAc sugar sequence [113], but lectin binding is notoriously promiscuous in practice. The precise mechanism that enables isolated cells from *C. elegans* to adhere to PNA has yet to be identified, but it is likely enabled by non-specific binding of glycosylated transmembrane proteins to the PNA lectin. No group has directly tested the strength of these adhesions *in vitro*, but membrane tether-extrusion studies using AFM probes functionalized with PNA suggests these bonds can withstand tensile forces on the order of tens of picoNewtons [24]. To our knowledge this is the only available CAM for ensuring adequate cell attachment, as all other traditional CAMs used in cell culture experiments have resulted in weakly attached cells that can easily be released from substrates by gentle mechanical titration. The fact that PNA has not been previously used as a CAM in protein micropatterning experiments we were further interested in testing if the available methods for micropatterning were generalizable to PNA and isolated cells from *C. elegans*. However, we found that this non-specific binding posed a greater limitation than expected.

Our original experiment goals included being able to add a second CAM in addition to the PNA required for cell adhesion. We chose EHS-laminin since it is widely used, commercially available for purchase, and highly conserved. We also did not expect that cells would form attachments to laminin in cultures, based on reports from others [40, 84, 92, 114]. We were surprised to see cells attaching to the laminin in the laminin-PNA grids. One potential explanation is that PNA binds to laminin patterns (Figure 3.6), leading cells to attach to PNA as usual and accounting for the increased meandering of TRN neurites observed in this case. Laminin is well known to act as a guidance cue for neurite outgrowth in multiple systems, however. Thus, it is also possible that the cellular phenotypes observed on laminin-PNA grids could be the result of laminin acting as a restrictive yet instructive cues. Nevertheless, these findings do inform us of the limits of micropatterning. With dual-CAM patterning there is always the possibility of the two proteins of interest

interacting. Even when using fluorescently tagged proteins to visualize the degree of colocalization it is technically challenging to draw certainties regarding the absence of these potential interactions.

3.3.2 Future Directions

The need for a dual-CAM micropatterning approach warrants follow-up work to reduce the potential for interactions between the two CAMs. PNA seems to have a high affinity for EHS laminin in culture. In preliminary experiments, we attempted to block this interaction using galactose in soluction to block potential PNA-laminin binding sites. Results from those experiments were inconclusive with regards to if the interaction was truly being blocked and if cells would still attach to the PNA. It would be interesting to follow up with these experiments and see if it is a viable approach. An alternative path would be to use a biotin-streptavidin patterning process to reduce the likelihood of interaction. Independent of the ultimate path towards a dual-protein patterning process, an important future direction would be to purify *C. elegans* ECM proteins for patterning ECM.

3.4 Materials and methods

3.4.1 Pattern Design

All patterns were designed using the open-source software Inkscape with a canvas size of 1824x1140 pixels. Since the physical size of the patterns depends on the magnification of the objective, the calibration of the PRIMO tube lens, and the positioning of the focal plane during patterning, pattern specifications are made in pixel units. On average, by my records, each pixel correspond to 0.27 μm at the time of exposure. The final relationship between pattern features and the LIMAP pixel size can vary greatly with patterning quality.

3.4.2 PDMS Stencil Fabrication

PDMS stencils were drawn in the freely-available Silhouette software package which interfaces with the Silhouette Cameo die-cut machine (Stanford Nanofabrication Facilities). The Cameo tool

cuts the custom design from a PDMS sheet that is loaded into the Cameo. Stencils were generated in batches of 24 and stored until required for use. PDMS stencils could also be generated by similar craft die-cut machines such as a Cricut.

3.4.3 LIMAP patterning

Multiwell, tissue culture plates were handled with aseptic technique at all steps to minimize contamination. Plates were oxygen plasma treated at Stanford Nanofabrication Facilities (1 min, FWD power 25, REFL 0), prior to processing according to the workflow described earlier. Once prepared, substrates were patterned using an Alvéole PRIMO system and its Leonardo software on an inverted Leica microscope and Hamamatsu Orca camera. Substrates were prepared the day before patterning. Patterns were fabricated the day before seeding cells and kept at ambient conditions, flooded with PBS, until cell seeding the next day. PBS was mostly removed except for 20 µl that was kept within the PDMS stencil. This was exchanged for cell culture media just before removing the PDMS stencil. A 200 µl droplet of cells in solution were immediately and slowly added to the remaining liquid above the patterns to form a stable bubble. This bubble sat undisturbed after replacing the plate lid for 15 minutes before the well was gently filled with media at the side of the well. This procedure ensured that sufficient numbers of cells were positioned directly over the pattern. This is critical for TRN experiments because (1) TRNs are only 1% of total cells, and (2) the cell bodies of TRNs are unlikely to travel large distances to "find" the PNA stripes .

3.4.4 Worm husbandry

GN753 pgSi116[Pmec-17::mNeonGreen::mec-4::tbb-2 3'UTR] II animals were used for all experiments shown here. Husbandry techniques used for these experiments to generate the 50,000-100,000 worms needed for cell culture prep were identical to those detailed in Chapter 2. Animals were kept as dauers until 2 weeks prior to experimentation. Animals were thawed from freezer stocks roughly every 6 months, to minimize the emergence of spontaneous mutations. Newly thawed stocks were maintained on NGM plates seeded with OP50 *E. coli*, expanded through three generations, with synchronization when the population reached the gravid stage for each generation.

Following the initial step, animals were grown on enriched peptone Agar plates with NA22 *E. coli*. Animals were maintained at 20°C.

3.4.5 Cell culture

Methods are identical to those detailed in Chapter 2. Embryos were harvested from GN753 animals and dissociated using standard techniques [88]. Cells were seeded immediately following dissociation in phenol red-free L-15 cell culture media without HEPES. Cultures were maintained in chambers without C02 regulation and kept at room temperature. Following cell seeding, the vented lids of the multiwell cell culture plate was sealed with parafilm to minimize evaporation and reduce contamination by environmental microbes.

3.4.6 Quantification of TRN morphology and mNG::MEC-4 expression

Images of TRNs were acquired and analyzed as described in Chapter 4. Briefly, the pgSi116 transgene encoding mNG::MEC-4 allows for full visualization of the neurite. Custom, supervised analysis code measures neurite length as the distance from where the axon initial segment meets the cell body and the distal tip of the straightened neurite. The neurite region is defined as the 1 μm space centered about this straightening line. Average Neurite Intensity is the average pixel intensity of all pixels within this region. All image analysis code for estimating neurite length is detailed in Appendices B and C.

3.4.7 Microscopy

All images were acquired on a compact inverted microscope (Keyence, Model BZ-X800) using either a Nikon 20x or 60x (oil immersion) objective. Imaging conditions were the same for all quantification experiments (60x oil immersion). Cells were kept within the micropatterned device throughout imaging, with the plate lid and original parafilm seal in place to minimize evaporation. Images were acquired in high resolution mode (0.127 μm/pixels) with exposure times of 1.5 ms for the mNG::MEC-4 imaging channel. Exposure times for the red imaging channel were adjusted depending on the quality of the patterns (brighter patterns require less exposure time).

Chapter 4

Laminin and nidogen determine MeT channel localization in *C. elegans* TRNs

4.1 Preface

In this chapter I present a manuscript that will be submitted for peer-review. My contributions as co-author include conceptualization, methodology development, software, validation, formal analysis, investigation, resources, data curation, drafting and editing text, and visualization. The intellectual framing of this manuscript is built on the scientific background detailed in Chapter 1, the *in vitro* data was collected using the protocols refined and tools developed in Chapters 2 and 3, in addition to quantitative image analysis tools. The details of this computational workflow are described in greater detail than was practical for the purposes of this manuscript in Appendices B, C, and D. The results presented here illustrate the utility of the methods developed in this thesis and how they have helped us build our understanding of MeT channel positioning *in vivo*.

4.2 Laminin and nidogen determine MeT channel localization

Title

Laminin and nidogen determine mechanoelectrical transduction channel localization in *C. elegans* touch receptor neurons

Authors

Alakananda Das[1], Joy A. Franco[2], Lingxin Wang[1], Dail Chapman[1], Lucy M. Wang[2], Beth L. Pruitt[3], Ellen Kuhl[2], Miriam B. Goodman[1*]

[1] Department of Molecular and Cellular Physiology, Stanford University, Stanford, CA 94305

[2] Department of Mechanical Engineering, Stanford University, Stanford, CA 94305

[3] Departments of Mechanical Engineering and Molecular, Cellular, & Developmental Biology, University of California Santa Barbara, Santa Barbara, CA

*Correspondence: mbgoodmn@stanford.edu

Summary

Cellular mechanosensation is transduced by ion channels that convert mechanical stimuli into electrical signals. Using touch receptor neurons (TRNs) in *Caenorhabditis elegans* nematodes, we show that the mechanoelectrical transduction (MeT) channel MEC-4 is anchored at discrete puncta sites along the TRNs *in vivo* but not in cultured TRNs *in vitro,* suggesting channel localization depends on factors present *in vivo*. We identified a null allele of the conserved extracellular matrix (ECM) protein nidogen where MEC-4 puncta distribution is altered in number and spacing between puncta. We discovered that nidogen regulates behavioral and electrical responses to touch. Nidogen associates with laminin networks, and we show that both nidogen and laminin co-localize with MEC-4 puncta. Lastly, using computational modeling we propose that tethering the TRN to ECM at discrete points may increase the sensitivity of mechanoelectrical transduction channels by creating localized areas of high strain energy density.

Keywords

Extracellular matrix, neuron, mechanoelectrical transduction channels, mechanosensation, touch

Introduction

Our senses of hearing and touch are made possible by specialized cells that rely on mechanoelectrical transduction (MeT) ion channels to convert mechanical stimuli into electrical signals. MeT channels allow for rapid communication between cells and extracellular structures, resulting in membrane depolarization within hundreds of milliseconds of stimulation. Some of these MeT channels are activated by membrane stretch in a force-from-lipids mode, others are activated by deformation of protein tethers in a force-from-filament mode (Katta *et al.*, 2015; Martinac, 2014). Intracellular or extracellular filament proteins can have direct roles in ion channel activation by acting as physical tethers (Hayakawa *et al.*, 2008; Hu *et al.*, 2010). In other cases, sensory functions can be indirectly affected by factors that regulate cellular morphology, protein trafficking, cell and matrix stiffness or the position of receptor proteins within cells. Subcellular localization of mechanosensory ion channels is especially important for localized force sensation. The TRPC7 channel localization to focal adhesions in primary rat embryo fibroblasts (Gopal *et al.*, 2015) is an example of a MeT channel localizing to regions of high stress formed by cell-matrix tethers.

For decades the transparent roundworm *C. elegans* has been used as a model system for mechanosensation as it has a highly stereotyped behavioral response to gentle touch that allows for rapid mutant screening. These screens led to the early identification of six neurons, the Touch Receptor Neurons (TRNs), that act as mechanoreceptors allowing the animal to sense low-threshold physical stimuli. This sensing process is mediated by the *mec-4* gene which encodes a degenerin epithelial sodium channel subunit, MEC-4 (O'Hagan *et al.*, 2005). Immunohistochemistry labelling of MEC-4 revealed that this protein forms distinct puncta along the full length of the neuron with minimal variation between animals (Cueva *et al.*, 2007). The punctate distribution of MEC-4 along the neurite is hypothesized to tune both TRN sensitivity and frequency selectivity. Increasing MEC-4 spacing through genetic mutations results in a decreased behavioral response to gentle touch (Petzold *et al.*, 2013). Recently, through combined electrophysiology and finite element modeling Katta et al. (Katta *et al.*, 2019) showed that the number of channel complexes recruited during mechanostimulation scales with the magnitude of the stimulus, demonstrating a physiological role for channel distribution in mechanosensation (Katta *et al.*, 2019).

Mechanosensory neurons rapidly transduce physical stimuli into neural impulses. The mechanisms by which MeT channels formed by MEC-4 detect or respond to these stimuli in such short time scales have yet to be elucidated. One leading hypothesis is that the MeT channels are tethered to the ECM that surrounds the mechanoreceptor process, either through indirect mechanical coupling or direct binding (Christensen and Corey, 2007; Krieg *et al.*, 2015; Ranade *et al.*, 2015). The dynamic range of the mechanoreceptor may be in part determined by the properties of the ECM (i.e., their mechanics are coupled), which is composed of proteins contributed by both the TRN and the hypodermal cell surrounding the TRN. In the case of direct binding, tethering to ECM proteins may aid in the initial positioning and further stabilization of MeT channels along the neurite process, potentially regulating mechanoreceptor's sensitivity (Katta et al., 2019). Several mutations in the extracellular domain of MEC-4 have been characterized that result in loss of touch sensation (Eastwood and Goodman, 2012), suggesting that these residues may be important for interactions with ECM proteins. Thus, elucidating the biophysical properties and mechanisms of MeT channel gating requires a thorough study of the role of ECM proteins, in channel localization and stability.

Genetic screens in *C. elegans* identified three ECM genes that specifically disrupt gentle touch sensation: *mec-5*, *mec-1* and *mec-9* (Gu et al., 1996). *mec-5* encodes a collagen, while *mec-1* and *mec-9* both encode proteins with an array of epidermal growth factor-like (EGF) domains and Kunitz (Ku) domains. A previous study has shown that MEC-4 puncta along the lateral TRN neurites are absent in *mec-5*, *mec-1* and *mec-9* mutants (Emtage *et al.*, 2004). It is unclear whether the touch insensitivity in these mutants is due to the loss of a filament tether in the ECM or overall loss of channels from the neurite. Additionally, these studies were

unable to address whether ECM proteins confer temporal and positional stability to the MEC-4 channels, as perturbation experiments eliminated all MEC-4 puncta and/or touch sensitivity (Du *et al*. 1996; Emtage *et al*., 2004). A later study, using neuronally enhanced feeding RNA interference to uncover nonviable genes involved in gentle touch sensation, identified several proteins that form adhesion/focal adhesion complexes, including integrin (*pat-2*), cadherin (*hmr-1*), paxillin (*pxl-1*) and laminin (*lam-2*) (Chen *et al*., 2015). A hypomorphic mutation in laminin (*lam-1*) was also previously reported to partially suppress TRN degeneration caused by *mec-4*(*u231*d) allele (Kao, Driscoll and Wadsworth, 1995, personal communication), suggesting a potential role of laminin in mechanosensation.

Laminin is a well characterized and conserved ECM protein important for the structural and functional integrity of basement membranes in all metazoans. Laminin is a heterotrimeric protein composed of laminin-α, laminin-β and laminin-γ subunits. The human genome encodes 5 laminin-α, 3 laminin-β and 3 laminin-γ proteins, which assemble into 16 different trimer configurations (Aumailley, 2013). Among these, Laminin-332 secreted by mouse epidermal keratinocytes was shown to inhibit sensory neuron branching behavior and suppress rapidly adapting type of mechanosensitive currents in dorsal root ganglion neurons, by preventing the formation of protein tethers necessary for current activation (Chiang et al., 2011). *C. elegans* has two laminin-α (LAM-3 and EPI-1), one laminin-β (LAM-1) and one laminin-γ (LAM-2) encoding genes, leading to only two possible trimer configurations which have distinct functions (Huang et al., 2003). Null alleles of laminin subunits are homozygous lethal and hypomorphic alleles that survive to adulthood show severe defects in tissue organization (Huang et al., 2003). For these reasons, laminin's role in modulating touch sensation is not well understood. *C. elegans* also express several other highly conserved laminin-binding proteins such as nidogen (NID-1), perlecan (UNC-52), syndecan (SDN-1), dystroglycan (DGN-1), fibulin (FBL-1) and agrin (AGR-1) (Keeley et al., 2020). Nidogen has been shown to form a stable complex with laminin in multiple species, and often acts as a cross-linker between laminin, collagen, and perlecan in the ECM (Hohenester and Yurchenco, 2013). This binding appears to be functionally conserved, as *Drosophila* laminin can bind to mammalian nidogen. Nidogen has also been implicated in *Drosophila* mechanosensation, where null mutations to the nidogen producing gene results in reduced behavioral response to vibrational stimuli (Wolfstetter *et al*., 2019). Single cell RNA sequencing data in the worm indicates that TRNs express nidogen (Lockhead *et al*., 2016, Cao *et al*.,2017), and is known to localize to the TRNs in a punctate fashion (Ackley *et al*., 2005) making it an interesting candidate in worm mechanosensation.

In this study, we attempted to understand how the different ECM proteins making up the complex structure of the mantle around TRNs, affect each other and the localization and function of MEC-4 channels. We screened labeled ECM proteins to find unknown candidate proteins that constitute the ECM around the lateral TRNs and found that conserved basement membrane proteins laminin and nidogen are present in a punctate distribution along the TRNs and co-localize with MEC-4 along ALMs in a MEC-5, MEC-1 and MEC-9-dependent manner. We showed that the nidogen, laminin and MEC-4 puncta are highly immobile with very slow diffusion rates of the molecules, suggesting that these proteins likely form complexes that are physically anchored in place. We confirm the hypothesis that the ECM is primarily responsible for channel stability, as time lapse imaging experiments of MEC-4 in isolated TRNs show puncta lack the positional stability observed *in vivo*. We also found that although laminin and MEC-4 puncta along ALMs are dimmer and much sparser in nidogen mutants compared to wild-type, touch response behavior is only mildly affected. Finally, by observing the distribution of individual components in the absence of others, we propose a model of how these different ECM proteins influence each other's spatial organization.

Results

The space between the lateral TRNs and the surrounding hypodermal cell is filled with a distinctive ECM known as the mantle (Hekimi and Kershaw, 1993). Touch associated proteins, MEC-1 and MEC-5 were

previously observed to localize in the mantle along the TRNs by fluorescence and electron microscopy (Cueva et al., 2007; Emtage et al., 2004). Another touch associated protein, MEC-9 is also thought to be present in the ECM along TRNs although so far there have not been any studies confirming its localization.

ECM proteins laminin and nidogen are present in a punctate distribution along TRNs

To better understand the organization of this specialized ECM and how it affects touch sensation, we performed a screen to identify other ECM proteins that constitute the mantle. We tested 29 *C. elegans* strains expressing fluorescent protein-labeled core basement membrane proteins or their putative transmembrane receptors under endogenous *cis* regulatory control and looked for their localization along the TRNs. These 29 transgenic alleles were all created using CRISPR-Cas9 gene editing and represent 26 proteins available for analysis. We focused our observations on the ALM neurons because no other neurons are nearby and the ALM cell bodies and axons are separated from the body wall muscles and other tissues known to express core basement membrane proteins. Using widefield fluorescence microscopy, 11 of 26 proteins appeared to decorate the ALM neurites either in a uniform or punctate fashion. We found Z of X proteins uniformly distributed along ALM: FBL-1 (fibulin), MIG-6 (papilin), UNC-52 (perlecan), HIM-4 (hemicentin), SAX-7 (L1CAM), SDN-1 (syndecan) and LET-805 (myotactin) (Figure 1)]. Similar observations were reported using overexpressed GFP fusions with HIM-4.

Three classic basement membrane proteins, LAM-1 (Laminin β-subunit), LAM-2 (Laminin γ-subunit) and NID-1 (nidogen), localized to puncta distributed along TRNs (Figure 1). Laminin is a heterotrimeric basement membrane protein composed of laminin-α, laminin-β and laminin-γ subunits. *C. elegans* has one laminin-β (LAM-1) and one laminin-γ (LAM-2) protein, but two laminin-α (LAM-3 and EPI-1) proteins. Neither laminin-α protein was detected along the ALM neurites or cell body when tagging the endogenous loci with mNeonGreen or mScarlet or by immunofluorescence (Supplementary Fig 1). Given that laminin is thought to function exclusively as a trimer, these findings leave open the question of which laminin-α is responsible for organizing this atypical laminin structure or raise the possibility that another protein substitutes for laminin-α in these puncta. Laminin and nidogen puncta were not obviously associated with other neurons or tissues such as the pharynx and gonad, where these proteins appear as continuous sheets. Collectively, these findings demonstrate that several, but not all core basement membrane proteins are repurposed to form a specialized, punctate ECM along the mechanosensory TRNs.

Laminin, Nidogen and MEC-4 co-localize to complexes that decorate TRNs

The distribution of laminin-β, laminin-γ, and nidogen puncta is reminiscent of the distribution of MEC-4 channel proteins in the TRNs (Katta et al. 2019; Chelur et al. 2002; Cueva et al. 2007; Emtage et al. 2004), suggesting that these proteins associate with MEC-4 channels *in vivo*. To investigate this question in detail, we developed dual labeled strains expressing mNeonGreen (mNG) and wormScarlet (wSc) labeled pairs of proteins. We found that laminin and nidogen puncta strongly co-localize with each other and with MEC-4 puncta (Figure 2A). To quantify co-localization of protein pairs, we computed the intensity correlation quotient or ICQ (Li et al. 2004). For reference, we computed ICQ puncta co-expressing mNG-tagged and wSc-tagged NID-1 in the proximal and distal neurite segments. Consistent with the expectation that laminin and nidogen co-assemble into a single complex, ICQ values for LAM-1/LAM-2 and NID-1/LAM-2 pairs were similar to those measured for the NID-1/NID-1 pair. Based on the high ICQ values for LAM-1/MEC-4 and LAM-2/MEC-4 pairs (Figure 2X), we infe2r that the MEC-4 ion channel protein co-assembles with laminin and nidogen. This association is not shared with PAT-2, an α-integrin co-expressed with MEC-4 in the TRNs. Analysis of the PAT-2/MEC-4 pair yields ICQ values close to zero, consistent with the observation that MEC-4, but not PAT-2 localizes to puncta in the ALM neurons (Figure 2B).

Next, we sought to determine the mobility of these channel-ECM complexes using two approaches: time-lapse imaging and fluorescence recovery after photobleaching (FRAP). In time-lapse images, we

observed that nearly all mNeonGreen-tagged MEC-4, LAM-2 and NID-1 puncta are immobile over a time course of several minutes. Rarely and only proximal to the ALM cell body, we observed MEC-4 puncta moving in an anterograde direction. As shown in Figure 2X, stationary MEC-4 puncta recovered very little of their fluorescence 5 minutes after bleaching. This is not a general property of the ALM neurite and unlikely to reflect photodamage since myristoylated-GFP expressed in the same cells recovers half of its fluorescence only 1 minute after bleaching. This comparison implies that individual MEC-4 proteins are not able to diffuse in the plasma membrane and may be anchored in place. Similar to MEC-4 puncta and consistent, laminin and nidogen bleached puncta also failed to recover over a period of 5 minutes (Figure 2 C). Taken together, the colocalization of MEC-4 with laminin and nidogen and our FRAP observations suggest that MEC-4 channels form complexes with the ECM proteins laminin and nidogen and, further, that these channel-ECM complexes are anchored in place within the TRN neurites.

Nidogen is necessary for channel function and behavioral touch response

To understand what role these ECM proteins, play in the mantle along the TRNs, we investigated whether loss of these proteins have any effect on the touch response behavior of the animals. Consistent with previous reports, we observed greatly reduced touch sensitivity in the ECM mutants *mec-5(u444)*, *mec-1(e1496)*, *mec-1(e1738)*, *mec-1(e1526)*, *mec-9(u437)* and *mec-9(u34)* (Du et al., 1996; Emtage et al., 2004). In the *nid-1(cg119)* allele we also observed a smaller decrease in touch sensitivity. The *sdn-1(zh20)* mutant did not show any difference in touch sensitivity compared to wild-type (Figure 3A).

In order to investigate the underlying mechanism of reduced touch sensitivity in the ECM mutants, we studied mechanoreceptor currents (MRC) in ALM neurons from both wild-type and ECM mutant worms. We used a slit-worm preparation and *in vivo* whole-cell patch clamp recording to measure the electrical responses to mechanical stimulation in ALM neurons (Katta et al., 2019). To identify ALM neurons in wildtype and mutant animals, we used the allele *uIs31* which expresses GFP under the control of *mec-17* promoter. Our results revealed that *nid-1(cg119)* mutants have reduced MRC amplitude compared to MRCs from ALM neurons of control worm strain (Fig 3B-D). Kinetics of MRC in *nid-1(cg119)* animals, induced from either step-protocol and trapezoidal-protocol showed no difference compared to wild-type animals (Supplementary Figure xx). As expected, no MRC was detected from ALM neurons in *mec-5(u444)*, *mec-9(u34)*, *mec-1(e1526)* mutants, which show a strong defect in touch response behavior. None of these mutations affected voltage-activated currents in ALM neurons (Supplementary Fig 3), indicating that the defect is specific to MRCs in these mutants.

ECM organization along TRNs is necessary for proper MEC-4 puncta distribution

The gene *nid-1* codes for the laminin-binding protein nidogen that is highly conserved across vertebrates and invertebrates. In *C. elegans*, nidogen has been previously shown to be important for axon guidance but not for basement membrane assembly (Kim and Wadsworth, 2000). Its role in touch sensation and MEC-4 puncta localization in TRNs has not been previously reported. To understand why *nid-1(cg119)* mutants have a lower mechanoreceptor currents compared to wild-type, we used the previously described *pgSi116* allele (*mec-17p::mNeonGreen::mec-4*) to visualize MEC-4 puncta along ALM neurons (Katta *et al.*, 2019). MEC-4 inter-punctum interval (IPI) values obtained from this strain are comparable to what was reported previously (Árnadóttir *et al.*, 2011; Chelur *et al.*, 2002; Chen *et al.*, 2015; Cueva *et al.*, 2007; Emtage *et al.*, 2004; Petzold *et al.*, 2013; Vásquez *et al*., 2014; Zhang *et al.*, 2004). Previously Katta *et al*., also reported that the labeled MEC-4 puncta in this allele were more closely spaced proximal to the cell body compared to the distal regions of the neurite. In this study, we observed fewer MEC-4 puncta in the *nid-1(cg119)* mutant animals compared to wild-type (Fig. 4A). We also found that the *nid-1(cg119)* mutants showed higher IPI values in both the proximal and distal segments of the ALM neurite, compared to wild-type (Fig. 4B). Null alleles of other putative laminin

binding proteins such as syndecan (*sdn-1(zh20)*), dystroglycan (*dgn-1(cg121)*) and fibulin (*fbl-1(hd43)*) also did not show any significant difference in MEC-4 puncta distribution compared to wild-type (Supplementary Figure 4A).

Integrins are another class of known transmembrane laminin receptors. We tested an alpha integrin mutant strain, expressing wild-type PAT-2 in muscles to circumvent larval lethality. ALMs in this strain showed moderate morphological defects in axon guidance, abnormal cell body position and ectopic branching. However, MEC-4 puncta similar to wild-type were present along the neurites (Supplementary Figure 4A). Laminin is an important basement membrane component and is required for the proper development of many different tissues (Huang *et al.*, 2003). Consequently, null alleles of laminin subunits often are embryonic lethal. We tested the laminin-α, *epi-1(gm57)* mutant, which showed severe morphological defects and low brood sizes. In this background, the animals that survived to adulthood showed severe defects in axon guidance, abnormal cell body position and ectopic branching (Supplementary Figure 4A), but MEC-4 puncta similar to wild-type were visible.

Although *in vivo* studies clearly point to an essential role of the ECM in the final positioning and stabilization of MEC-4 puncta, we questioned if certain aspects of the observed distribution might be cell autonomous. Aiming to identify cell-intrinsic determinants of MEC-4 expression *in vitro*, we also quantified MEC-4 localization in cultured TRNs isolated from dissociated *C. elegans* embryos (STAR Methods, Strange *et al.*, 2007). Cell fate for the pair of lateral TRNs (ALM and PLM) is determined early in development, prior to the formation of the animal's cuticle (Sulston 1983). Thus, *C. elegans* embryos can be easily dissociated to yield heterogeneous cultures of isolated cells that will attach to peanut agglutinin (PNA) treated surfaces and rapidly complete differentiation (Christen *et al.*, 2002). A subpopulation of these cells is known to express TRN-specific genes such as *mec-3* and *mec-17*, indicating the presence of terminally differentiated TRNs that possess wild-type neurites within 24 hours of cell-seeding (Zhang *et al.*, 2003). Further characterization of these isolated TRNs established that most of them will adopt an ALM-like fate (Zheng *et al.*, 2015; Lockhead *et al.*, 2016). By narrowing our analysis to the subset of cells which exhibited both unipolar neuron morphology and expressed mNG::MEC-4 driven under *mec-17* promoter, we were able to directly compare MEC-4 expression patterns *in vitro* with *in vivo* correlates. A single study had previously reported surface expression of MEC-4 in isolated cells, however high levels of non-specific staining and a lack of pre-staining incubation makes these results difficult to interpret alone.

[needs a rationale for using the patterns and to provide a smooth transition] We cultured the isolated cells on custom micropatterns of peanut agglutinin to facilitate cell adherence while controlling morphology, but otherwise lacking the structured ECM around neurites that is present *in vivo* (STAR Methods, Strale *et al.*, 2016). We confirmed this by showing that wormScarlet labeled NID-1 did not localize in a punctate distribution along the neurite as observed *in vivo*. To quantify mNG::MEC-4 puncta distribution we imaged the cells at 24 hours *in vitro* and parsed these through custom analysis code (STAR Methods, Github). High levels of mNG::MEC-4 not associated with puncta allowed us to fully visualize neurite morphology including, in some cases, lamellipodia and growth cones. We verified that this background signal could be used as a proxy for full neurite morphology by repeating these experiments with cells expressing a red cytosolic marker. These experiments confirmed that the diffuse mNG signal perfectly colocalized with the red cytosolic reporter and that we would be able to calculate full neurite length directly using background mNG signal. We observed that these isolated TRNs were typically shorter than their *in vivo* counterparts, regardless of time in culture, and possessed far fewer puncta that were spaced farther apart than ALMs *in vivo* (Fig 4C,D). For a subset of experiments, we imaged these cells at both 24 and 48 hours *in vitro* and despite noting a clear lack of stability in puncta location, neurite morphology and distributions of MEC-4 puncta did not appreciably change between those time points.

In absence of wild-type ECM mantle, MEC-4 puncta remain sequestered in vesicles

In cultured TRNs, we observed that the MEC-4 puncta are comparatively more mobile, with many puncta showing sporadic movement and few puncta showing longer range stability. Time-lapse imaging experiments showed MEC-4 puncta moving at speeds consistent with active transport. To test this hypothesis we used the wyIs348 allele which expresses an mCherry::RAB-3 fusion protein, an established *C. elegans* vesicle marker (Maeder et al., 2013), under the TRN specific promoter *mec-17p*. We generated a transgenic animal expressing both *wyIs348* and *pgSi116* alleles, allowing us to visualize the colocalization of MEC-4 with RAB-3. We cultured cells isolated from the embryos of these animals and observed high expression levels of mCherry::RAB-3 in all mNeonGreen::MEC-4 expressing cells. The mCherry (mCh) signal appeared most strongly in the cell body with punctate expression along the neurite. These puncta varied in size, with some appearing as elongated ellipses oriented in the direction of the neurite, and others appearing small and spherical at the edges of the lamellipodia. We consistently observed colocalization between MEC-4 and RAB-3 in all imaged cells and sought to quantify this colocalization using custom code to calculate Pearson's Correlation Coefficient (PCC) between the mNG and mCH signals (STAR Methods, Github). With this approach we found that most MEC-4 puncta had a PCC greater than .5 when compared to the corresponding mCH image, suggesting that the majority of MEC-4 puncta observed in isolated TRNs are sequestered in vesicles.

The observation that MEC-4 colocalized with the RAB-3 vesicle marker in isolated TRNs motivated us to re-examine MEC-4 puncta mobility *in vivo* using this same transgenic line.
In time-lapse images of mNeonGreen::MEC-4 in ALM neurons, we observed that the majority of MEC-4 puncta are immobile over a time course of several minutes. On rare occasions we observed MEC-4 puncta moving bidirectionally along the neurite *in vivo*, with frequent pauses and reversals in direction. These moving puncta are more frequently observed closer to the cell body. We observed the co-localization of some moving MEC-4 puncta with RAB-3 labeled vesicles in wild-type animals (Fig 5), confirming that a small subset of MEC-4 channels are present in vesicles along the TRN.

Although MEC-4 puncta were severely reduced in *mec-5(u444)*, *mec-1(e1496)*, *mec-1(e1738)* and *mec-9(u437)* backgrounds as expected from previous studies (Emtage et al., 2004), we often observed a few bright mNG::MEC-4 labeled particles in these animals, usually closer to the cell body (Supplementary figure 5). In addition, we also found that in another *mec-1* allele (*e1526*), dim MEC-4 puncta were often visible in the proximal ALM segment but rarely in the distal ALM segment (Supplementary figure 5). To verify whether these puncta in the ECM mutants represented MEC-4 channels that were sequestered in vesicles, we looked at mCherry::RAB-3 and mNG::MEC-4 co-localization in wild-type and in these mutants (Supplementary figure 5). The average number of RAB-3 labeled vesicles per neuron and the average number of RAB-3 labeled vesicles that also contained mNG::MEC-4 was not significantly different between wild-type and ECM mutant animals. In wild-type animals, many of the bright and discrete mNG::MEC-4 fluorescence signal peaks that was previously classified as puncta, co-localized with the RAB-3 vesicle marker. In contrast, the majority of the punctate signal in the *mec-5(u444)*, *mec-1(e1738)*, and *mec-9(u437)* co-localized with mCherry::RAB-3 vesicle marker. This further reinforces the idea that in these ECM mutants, MEC-4 channels are unable to localize in a punctate manner in the plasma membrane although they appear to be trafficked normally in vesicles along the neurite. In contrast in the *mec-1(e1526)* and *nid-1(cg119)* backgrounds, only a small number of the punctate mNG::MEC-4 signal colocalized with mCh::RAB-3, indicating that in these animals, a significant fraction of MEC-4 channels were able to assume bona-fide punctate localization at the plasma membrane (Fig 5).

Laminin and nidogen form MEC-1, MEC-5 and MEC-9-dependent puncta along TRNs

We then tested whether the laminin and nidogen puncta distribution along TRNs are dependent on other ECM proteins or MEC-4. LAM-1, LAM-2 and NID-1 puncta are present in *mec-4(u253)* animals indicating that MEC-4 channels are not required for punctate laminin and nidogen localization along TRNs (Figure 6). In contrast, LAM-1, LAM-2 and NID-1 puncta are completely lost in the *mec-5(u444), mec-1(e1496)* and *mec-9(u437)* animals. In *mec-1(e1738)* and *mec-9(u34)* animals, we observed faint uniform LAM-2 and NID-1 fluorescence along TRNs instead of puncta.

Surprisingly, in the *mec-1(e1526)* mutant background where MEC-4 puncta are dimmer and are concentrated in the proximal segment, laminin and nidogen puncta are present all along the neurite (Figure 6). Proximal to the cell body, the dim MEC-4 puncta co-localize with NID-1 and LAM-1 puncta. This is reflected in the ICQ values for LAM-1/MEC-4 and NID-1/MEC-4 in *mec-1(e1526)*, which are lower in the distal segment compared to the proximal segment, and both of which are lower than that in the wild-type background. The ICQ values for NID-1/LAM-2 in the *mec-1(e1526)* background is similar to wild-type background in both proximal and distal segments (Figure 6).

Laminin requires nidogen to form puncta along TRNs

In *nid-1(cg119)* animals, LAM-1 and LAM-2 puncta were dimmer and spaced further apart compared to wild-type, in a manner like what we observed for MEC-4 puncta in *nid-1(cg119)* animals (Figure 7). We still observed weak co-localization of MEC-4 and LAM-1 puncta in *nid-1(cg119)* animals with low ICQ values, since the fluorescence signal from laminin puncta is barely visible above the signal from other tissues in the background.

Given that nidogen is known to bind laminin with high affinity, we asked whether nidogen is still able to localize in a punctate distribution along TRNs when laminin is disrupted. To investigate this, we attempted to knock down *lam-2* expression in animals expressing LAM-2::mNG and NID-1::wSc. We transferred a synchronized population of L1 arrested animals into fresh plates containing bacteria expressing *lam-2* RNA and tested these worms when they reached adult stage after 3 days. We monitored the change in LAM-2::mNG fluorescence intensity or localization pattern of LAM-2::mNG in various tissues to determine if the laminin was effectively depleted. Animals fed with bacteria expressing *lam-2* RNA, showed a dramatic reduction in overall LAM-2::mNG fluorescence level compared to control animals and were sterile, indicating that the treatment was effective at depleting LAM-2::mNG levels in most tissues. However, upon closer inspection, we still observed faint LAM-2::mNG puncta along TRNs. In these animals, NID-1::wSc, which appears as a uniform layer in basement membrane around many tissues in control animals, mislocalized into large aggregates, but there was no change in the NID-1::wSc puncta along TRNs in fluorescence intensity or distribution.

To achieve a greater degree of laminin depletion, we next transferred L4 stage animals on plates containing RNA expressing bacteria and tested their progeny after 4 days. Under this treatment condition, the *lam-1* and *lam-2* RNAi fed worms failed to generate any viable progeny and thus could not be tested. The first-generation progeny of *epi-1* RNAi fed worms survived to adulthood but showed severe morphological defects characteristic of *epi-1* mutants, such as smaller body size, sterility due to lack of gonads. LAM-2::mNG failed to be secreted from the cells in many tissues due to a lack of laminin alpha subunit, and appeared as aggregates within cells, a phenotype which was reported earlier (Huang et al., 2003). In these worms the TRNs showed aberrant morphologies similar to *epi-1(gm57)* animals. LAM-2::mNG signal along TRNs were faint and difficult to monitor due to high background fluorescence from large cellular aggregates in neighboring cells and the misshapen neurites going in and out of the focal plane. However, the NID-1::wSc as well as mNG::MEC-4 puncta still appeared normal.

Discussion

Discussion

Outside-in mechanosignaling is a ubiquitous physiological function required for sustaining homeostasis with the environment at the cellular, tissue, and organism levels. This rapid form of mechanotransduction relies on the faithful encoding of mechanical stimuli by MeT ion channels distributed throughout mechanosensitive cells. This study brings to light key information about the conserved molecules that are necessary for positioning the MEC-4 MeT channel along the TRN neurite to ensure precise alignment with the surrounding ECM. *C. elegans* have a diverse adhesome that shares similarities with many other species, but this study shows that only a subset of these molecules is involved in the mechanosensory process. In our screen for ECM proteins that localize along the TRNs, we identified several common basement membrane proteins such as laminin (LAM-1, LAM-2), nidogen (NID-1), perlecan (UNC-52), hemicentin (HIM-4), syndecan (SDN-1) and fibulin (FBL-1). Among these, only laminin and nidogen assumed a punctate distribution along TRNs, while the others showed a uniform distribution. Although the punctate localization of NID-1 along TRNs was previously reported, we show for the first time that the nidogen and laminin puncta along TRNs co-localize with MEC-4 puncta in a robust and repeatable manner with little to no-variation. Mutations to genes encoding integrin (pat-2), hemicentin, dystroglycan (dgn-1), syndecan and fibulin did not alter MEC-4 puncta distribution, although TRN attachment defects were observed in the absence of PAT-2 and HIM-4. By performing a wide-scale quantitative analysis of thousands of puncta we identified a previously unreported role for NID-1 in directing the positioning of MEC-4 and laminin puncta surrounding the TRN neurite. Using FRAP we demonstrate that these proteins are not freely diffusing, which suggests that together the proteins could form a stable complex that tunes the mechanosensitivity of the TRN. This shift in dynamic range in the absence of NID-1 manifests both in electrophysiology recordings and in behavioral response to touch. These findings contribute to the broader model of the molecular mechanisms that enable force transduction between the ECM and MeT ion channels.

The temporal and spatial stability of MEC-4 channels has not been previously reported. The findings from our parallel *in vivo-in vitro* analysis suggest that the ECM is necessary for the positional stability of MEC-4 in the plasma membrane. Although extensive time lapse imaging of isolated TRNs was limited due to low signal intensity, we observed numerous cases of MEC-4 puncta moving with speeds on par with those of active trafficking. We also noticed that even those puncta that appeared stationary in position were subject to small, cyclic lateral motility reminiscent of Brownian motion. We did not observe a similar trend in our *in vivo* measurements. The MEC-4 puncta measured *in vivo* are highly immobile over a span of several minutes. The puncta that did move on rare occasions co-localized with RAB-3 labeled vesicles indicating that the channels in these moving puncta have not yet reached the plasma membrane.

We have not observed either of the two known laminin-α proteins, LAM-3 and EPI-1, along TRNs in strains where fluorescent protein encoding sequences were inserted at the 3' end of these genes at the endogenous loci. In the same strains, LAM-3 and EPI-1 was observed in other tissues where it was previously detected by immunofluorescence (Huang et al., 2003). These animals also appeared otherwise normal in morphology and reproduction. This indicates that insertion of the fluorescent protein tag at the C-terminus did not affect the secretion and function of laminin-α subunits. LAM-3 and EPI-1 are orthologs of vertebrate laminin-α1/α2 and laminin-α3/α5, respectively. In human keratinocytes, the laminin-α5 chain has been shown to be proteolytically processed between the third and fourth globular domains at its C-terminus (Senyürek et al., 2014). In an earlier study, about one-third of the laminin molecules secreted by a human malignant cell line (JAR) derived from gestational choriocarcinoma was found to contain a smaller alpha subunit that was cleaved by limited proteolysis (Peters et al., 1985). EPI-1 is the predominant laminin-α subunit secreted by the epidermis (Huang et al., 2003) and hence is most likely responsible for the secretion of LAM-1 and LAM-2 into the extracellular space surrounding the TRNs. The absence of EPI-1::wormScarlet or EPI-1::mNeonGreen signal around the TRNs may be due to proteolytic cleavage at the C-terminus.

Previously, a hypomorphic mutation in laminin-β (*lam-1(rh219)*) was reported to partially suppress TRN degeneration caused by *mec-4*(*u231*d) allele (Kao G, Driscoll MA, & Wadsworth WG (1995). *Worm Breeder's Gazette, 14(1)*, 96.). This observation may be explained by the inability of MEC-4 channels to localize correctly along the TRNs. However, in the *epi-1(gm57)* background MEC-4 puncta appear normal even though the neurite and the animal as a whole has severe morphological defects. The *gm57* allele is unmapped and the nature of mutation is unknown. It is possible that the protein product of this allele is sufficient for MEC-4 puncta formation but not for normal basement membrane integrity. Knocking down *epi-1* by RNAi also generated similar results. However, since complete loss of laminin along TRNs could not be achieved by RNAi treatment, the residual laminin may be sufficient for directing MEC-4 puncta distribution.

Laminin subunits trimerize through their respective coiled coil domains. A disulfide bond between cysteine residues at the C-termini of the beta and gamma subunits is crucial for the laminin beta-gamma dimer formation (CITATION). In a recent report it was shown that CRISPR-Cas9-mediated labelling of the C-terminus of laminin-β1 in human lung adenocarcinoma cell line A549 leads to secretion inhibition (Shaw et al., 2020). In LAM-1::wSc strain, adding a fluorescent protein at the C-terminus of LAM-1 (laminin-β) likely interfered with the disulfide bond formation and secretion, due to which homozygous animals expressing endogenously tagged LAM-1 showed laminin mutant phenotypes (widespread defects in tissue organization and sterility), although heterozygotes bearing a single unlabeled copy appeared normal. The homozygous LAM-1::wSc animals also showed defects in TRN morphology similar to *epi-1(gm57)* and *epi-1* knockdown by RNAi, but MEC-4 puncta distribution appeared normal.

Nidogen is known to interact with many basement membrane proteins, including laminin, perlecan, fibulin and collagen. The high affinity binding site between nidogen and laminin-γ1 has been very well characterized structurally and biochemically and has been narrowed down to the nidogen G3 domain and the laminin-γ1 LEb3 domain, also known as the γ1III4 domain (). Deleting this domain in mouse embryonic stem cells by homologous recombination did not impair laminin heterotrimer assembly and secretion, but the embryoid bodies showed impaired deposition of nidogen into basement membrane like structures. Furthermore, mice harboring selective homozygous deletion of the γ1III4 domain were shown to die immediately after birth, with renal agenesis and impaired lung development (Willem *et al*., 2002).

Laminin is one of the earliest basement membrane components to be deposited during embryogenesis, being first detected by the end of gastrulation (Huang et al., 2003), i.e., 330 minutes post fertilization (Hall, D.H., Herndon, L.A. and Altun, Z. Introduction to *C. elegans* Embryo Anatomy. In *WormAtlas* (doi:10.3908/wormatlas.4.1). In contrast, the *mec-5*, *mec-9* and *mec-1* gene products do not appear until 450 minutes post fertilization, during the elongation phase of embryogenesis (CITATION). Ordinarily this would imply that laminin localization should not depend upon the presence of MEC-5, MEC-1 and MEC-9 proteins. However, we observed that laminin and nidogen was either completely undetectable or lost its punctate pattern in *mec-5*, *mec-1* and *mec-9* null mutants. This implies that crucial remodeling steps occur in the ECM around the TRNs that lead to the punctate localization of laminin. The Kunitz domain containing proteins, MEC-1 and MEC-9 are especially interesting in this regard since Kunitz domains have been shown to possess protease inhibitor activity. In fact, a missense mutation in the third Kunitz domain of MEC-9 (*mec-9(u34)*) shows faint uniform laminin and nidogen fluorescence around TRNs instead of distinct puncta. In contrast, in the *mec-1(e1526)* animals, which bear a missense mutation in the last Kunitz domain, laminin and nidogen puncta are present but are no longer co-localized with MEC-4 puncta. These observations suggest that MEC-1 might be a crucial protein linking laminin and MEC-4. In the future it would be interesting to investigate the roles of ECM remodeling proteases in the punctate localization of laminin along TRNs.

STAR Methods

***C. elegans* strain construction and maintenance**

A complete list of all *C. elegans* strains used in this study with their respective genotypes and provenance is available in Table 1. Animals were maintained at 20 °C on NGM agar plates seeded with *E. coli* OP50 unless otherwise mentioned. Insertion of fluorescent protein encoding DNA at endogenous loci, or single copy insertion of transgenes at MosSCI insertion sites was performed using Cas9-directed homologous recombination, according to self-excising cassette (SEC) method described in Dickinson et al., , 2015 (PMID: 26044593). For each strain, we injected a mixture of 50 ng/ul Cas9-sgRNA plasmids, 50 ng/ul repair template plasmid, 5 ng/ul pCFJ104 (myo3p::mCherry) and 2.5 ng/ul pCFJ90 (myo-2p::mCherry) into the gonads of 40-60 young adult N2 animals. Injected animals were singled and allowed to lay eggs for 2-3 days at 25 °C in the absence of selection. 500 ul of 5 mg/ml hygromycin solution was added to each plate, and the plates were returned to 25 °C for 4-5 days. Candidate knockin animals were dominant roller [*sqt-1(e1350)*] worms that survived hygromycin treatment and lacked red fluorescent extrachromosomal array markers. To excise the SEC, we heat-shocked plates containing ~6 L4 rollers each at 37 °C for 75 minutes, then grew the animals at 20 °C for 3-4 days. Adult wild-type animals (worms that lost both copies of the SEC) were singled and successful genome editing was verified by visualizing fluorescence and PCR genotyping.

[Describe gene editing method used for del-LE and del-G3 mutants]

All strains created in this lab for this study are available upon request.

RNA Interference

RNAi constructs for *lam-1*, *lam-2* and *epi-1* were obtained from the Ahringer library and that for *lam-3* was obtained from the Vidal library (Source Bioscience). The L4440 empty vector was used as a negative control. All RNAi experiments were performed using the feeding method (Kamath et al., 2003; Rual et al., 2004), in 6-well plates containing NGM agar supplemented with 25 ug/ml carbenicillin and 0.5 mM IPTG. Plates were poured 3-7 days prior to seeding bacteria on them. RNAi bacteria were streaked from glycerol stocks onto LB agar plates containing 25 ug/ml carbenicillin and 10 ug/ml tetracycline. Colonies from these plates were picked to grow in liquid culture in Luria Bertani (LB) media containing 25 ug/ml carbenicillin for 16 h at 37 °C. The cultures were then pelleted by centrifugation at 4000 rpm for 10 minutes in a swing bucket rotor ([insert rotor name]) in a [insert make/model of centrifuge], and the pellets were resuspended in fresh LB media containing 25 ug/ml carbenicillin. 150 ul of this concentrated bacterial culture was pipetted at the center of the agar in each well of the 6-well plate. For co-depletion experiments, the relevant bacterial cultures were mixed at a 1:1 ratio before they were seeded on plates. After seeding, the plates were incubated for 24 hours at room temperature to induce dsRNA expression, before adding worms. *lam-1* and *lam-2* RNAi feeding experiments were performed using synchronized L1 worms, which were placed on RNAi plates and allowed to feed for 72 h at 20 °C, before imaging. *epi-1* and *lam-3* RNAi feeding experiments were performed by placing 2 L4 stage worms in each well, and the progeny of these worms were imaged after 96 hours of incubation at 20 °C.

Microscopy

Worms were immobilized with 5 mM Levamisole dissolved in M9 buffer and mounted on agarose pads (5% agarose dissolved in M9). Images of ALM neurons were acquired on Keyence BZ-X800 inverted epifluorescence microscope with a 40x or 60x objective (Nikon Plan Apo 40x/1.0, Nikon Plan Apo lambda 60x). For FRAP experiments, worms were mounted similarly, and images were acquired on a Leica SP8 inverted laser scanning confocal microscope at Stanford's Cell Science Imaging Facility at 40X magnification. Pre-bleach images were acquired at 1 second intervals for 10 frames. Post-bleach images were acquired at 5 second intervals for mNG::MEC-4 and 1 second intervals for myr::GFP, for 60 frames.

Puncta analysis

For analyzing puncta distribution along the entire ALM neurite, overlapping images captured automatically on the Keyence BZ-X800 inverted epifluorescence microscope were seamlessly stitched together using the Keyence proprietary image stitching software. In the stitched images ALM neurites were traced as a segmented line (line parameters: width = 20 pixels, spline fit) and straightened using the built-in 'Straighten' function in ImageJ. These straightened images were read by a custom Python script, which extracted the fluorescence intensity values along the neuron after background subtraction. Puncta positions were calculated using the 'find_peaks' functions from the scipy library. The full script is available on Github (link to github).

FRAP analysis

For analyzing FRAP data between wild-type and mutant animals, ALM neurites from time-lapse fluorescent images were manually traced using the segmented line tool in ImageJ, to generate a 21-pixel wide straightened time lapse image sequence of the neurite as well as a kymograph. The kymographs were first inspected for overall movement of the neurite during acquisition as well as the presence of moving puncta and those samples were excluded from further analysis. In the remaining images, the bleach area was automatically detected by a custom Python script by finding the region of highest change in fluorescence intensity between the last two pre-bleach frames and the first two post-bleach frames. A region outside the neurite was selected for background subtraction. The average fluorescence intensity in the bleached region after background subtraction and normalisation to the overall photobleaching during acquisition was scaled in the range 0-1 according to the following equations:

$$I_{FRAP_{norm}}(t) = \frac{\frac{I_{bleach}(t)}{I_{bleach_{pre}}}}{\frac{I_{cell}(t)}{I_{cell_{pre}}}}$$

$$I_{FRAP_{norm_{scale}}}(t) = \frac{I_{FRAP_{norm}}(t) - I_{FRAP_{norm}(t=0)}}{I_{bleach_{pre}} - I_{FRAP_{norm}(t=0)}}$$

Where,

$I_{bleach}(t)$= Average intensity of the bleached area after background subtraction as a function of time.

$I_{bleach_{pre}}$= Average pre-bleach intensity of the bleached area after subtracting background intensity.

$I_{cell}(t)$= Average intensity of the cell after background subtraction as a function of time.

$I_{cell_{pre}}$= Average pre-bleach intensity of the cell after subtracting background intensity.

$I_{FRAP_{norm}(t=0)}$= Normalized FRAP intensity at the first frame after photobleaching (t=0).

The intensity curve during the recovery phase was fitted to the equation, $I = A(1 - e^{-B/t})$ where the parameter A denotes the mobile fraction, and the parameter B denotes the time constant of recovery. The half time of recovery was calculated using the equation, $t_{1/2} = -\frac{ln(0.5)}{B}$. The diffusion coefficient was calculated according to the equation, $= 0.225\frac{w^2}{t_{1/2}}$. The full script is available on Github (link to github).

Co-localization analysis

For analyzing co-localization of a pair of mNeonGreen and wormScarlet labeled proteins along the ALM neurite, multichannel overlapping images were captured automatically on the Keyence BZ-X800 inverted epifluorescence microscope and seamlessly stitched together using the Keyence proprietary image stitching software. In the stitched images ALM neurites were traced as a segmented line (line parameters: width = 20 pixels, spline fit) and straightened using the built-in 'Straighten' function in ImageJ. These straightened images

were read by a custom Python script, which extracted the fluorescence intensity values for each channel along the neuron after background subtraction. Intensity correlation quotient (ICQ) is defined as the ratio of the number of pixels where the product of difference from mean for the red and green channels are positive, to the total number of pixels subtracted by 0.5. As a consequence, the ICQ varies from 0.5 (co-localisation) to -0.5 (exclusion) while random staining and images impeded by noise will give a value close to zero. The full script is available on Github (link to github).

***C. elegans* touch assay**

For measuring touch response of *C. elegans* strains, synchronized populations of well-fed young adult animals were used. Mechanical stimulus was delivered by touching the worm with an eyebrow hair glued to a toothpick. Each worm was touched 10 times, alternating between the anterior and posterior regions of the worm body. A positive response was recorded if the worm reversed its direction of movement. Experiments were performed blinded to genotype in groups of four of five strains including N2 as positive control and TU253 (*mec-4(u253)*) as negative control in each group. For a single experiment, touch responses from 25 animals were recorded from each strain and each experiment was repeated on three different days.

Electrophysiology experiments

Mechanoreceptor current (MRC) was acquired from ALM neurons in TU2769, GN932, GN985, and GN986 worm strains. Electrophysiology recording extracellular solution contained (in mM): NaCl (145), KCl (5), $MgCl_2$ (5), $CaCl_2$ (1), and Na-HEPES (10), adjusted to a pH of 7.2 with NaOH. 20 mM of D-glucose solution was added to external recording solution before using the solution and this will bring the osmolarity to about 325 mOsm. The intracellular recording solution constrained (in mM): K-gluconate (125), KCl (18), NaCl (4), $MgCl_2$ (1), CaCl1 (0.6), K-HEPES (10), and K_2EGTA (10), and adjusted to a pH of 7.2 with KOH. 1 mM of sulforhodamine 101 (Invitrogen) was added to intracellular solution before using to help visualize whether the neuron was successfully recorded.

For electrophysiology experiments, EPC-10 USB amplifier controlled by Patchmaster software (version 2x90; HEKA/Harvard Biosciences) was used to regulate cell membrane potential and control the mechanical stimulator. Liquid-junction potential was corrected (-14 mV) in this study. Analoues data was filtered at 2.9 kHz and digitized at 10 kHz.

Mechanical stimulation

Mechanical stimulation was obtained using an open-loop system adapted from the piezoelectric stack system with a photodiode motion detector described by Peng and Ricci (2016) in their study on hair cells. Details of this mechanical stimulation system was introduced by Katta S et al., 2019. Generally, this mechanical stimulation system was controlled by HEKA EPC-10 PLUS amplifier and Patchmaster software. Mechanical stimulation signal was filtered at 2.5 kHz in this study.

MRC was induced by both step and trapezoidal stimuli. Step protocols had a 300-ms hold at the commanded indentation with an interstimulus interval (ISI) of 1 s. Trapezoidal protocol had a 300-ms hold and 2-s ISI.

Embryonic dissociations and cell culture conditions

All cell culture procedures were performed at the bench, outside of a biosafety cabinet, using aseptic technique. Cells were isolated from embryonic dissociations using standard methods (Strange *et al*., 2007, Sangaletti and Bianchi, 2013) with some variation. Animals were kept in the dauer phase until ~2 weeks prior to collection. Dauer animals were chunked to NGM plates until the gravid stage. They were subsequently lysed with strongly basic hypochlorite solution and their progeny seeded on enriched peptone plates with NA22 *E. coli*. This process was repeated for two additional generations prior to collection for embryonic dissociation. At

the time of dissociation, gravid adults were collected from the plates and repeatedly rinsed to remove excess bacteria. Following rinsing animals were lysed using strongly basic hypochlorite solution and resulting embryos were rinsed 3X with pH adjusted buffer before being isolated with a sucrose gradient. Following the separation of embryos from lysis remnants, the embryos were incubated with 25 U/ml chitinase while rocking until ~80\% of eggshells had been digested (~50 min). At this point embryos were repeatedly rinsed and then underwent mechanical dissociation with a 21 and then 25-gauge needle. Cells were filtered at 5 um to remove debris and then rinsed multiple times to ensure that any potential endogenous ECM proteins were removed. Cells were suspended in L-15 cell culture media without HEPES or phenol red, seeded onto custom micropatterns of peanut lectin, and stored at normal temperature with atmospheric gases.

Fabrication of protein patterns for cell cultures

Micropatterned substrates were prepared using aseptic technique. Custom patterns of 8-pixel wide lines were designed in Inkscape using a canvas size of 1824x1140 square pixels. Custom PDMS stencils for retaining reagents were designed using Silhouette software and cut from a thin PDMS sheet using the Cameo die cut machine. Micropatterns of peanut lectin were made using a custom protocol for light activated molecular adsorption (LIMAP) protein patterning, adapted from (Strale *et al*., 2016). Glass bottom six-well tissue culture plates were plasma treated with ionized atmospheric gas for one minute with 25 forward and zero reflected power (Plasmaetch P-50, Stanford Nanofabrication Facilities). PDMS stencils were positioned after plasma treatment. Following stencil placement, ~35 µl of 0.01% poly-l-lysine (PLL) solution was added to the microwell of the stencil and allowed to incubate for one hour, followed by incubation with freshly prepared mPEG-SVA in pH adjusted HEPES (100 µg/ml). Substrates were rinsed and allowed to air dry prior to the application of PLPP gel and pure ethanol. Surfaces were again allowed to dry prior to patterning. All patterns were generated using the Alvéole PRIMO system using standard procedures (25 ms exposure, 100% laser power, 50 mJ/mm^2 dosage). Patterns were incubated with PBS for at least five minutes prior to incubation with PNA (Alexa Fluor 594 conjugate, 500 µg/ml). All patterns were fabricated the day before cell seeding and kept hydrated with PBS until used. PDMS stencils were removed just prior to seeding.

Isolated TRN imaging

Imaging conditions for experiments quantifying puncta distribution were identical except for imaging time point. For the main results reported cells were imaged at 24 hours in culture. For a subset of experiments cells were imaged at 48 hrs. Cells were imaged on a Keyence BZ-X800 inverted epifluorescence microscope with a 60x oil immersion objective (Nikon Plan Apo lambda 60x) in high resolution mode (no binning), with GFP (1.5 sec exposure) and mCherry (1/60 sec exposure) filter sets. Brightfield images were also taken without binning, 1/150 second exposure. Widefield images were saved as 32-bit RGB .tiff digital files.

Mechanical modeling

Numerical simulations were performed using Abaqus/Standard. The neurite was modeled as a homogeneous elastic semicylinder with a diameter of 250 nm (Cueva et al, 2007) and Young's modulus of 6.3kPa (Krieg et al, 2017). Connection to the ECM was represented by a force tangential to the surface of the neurite. The magnitude of the force was calculated using the longitudinal strain at the connection location and a spring stiffness of 1 mN/m (Powers et al, 2012). This point force was distributed over a 0.5 um diameter region to eliminate singularities. The force decreased quadratically from the point of force application to the region border. In simulations with multiple connections, an inter-punctum interval of 3 um was used (Cueva et al, 2007). For both model geometries, x-symmetry boundary conditions were applied, and the z-displacement was fixed at the end farthest away from the indentation. In the indented geometry, z-symmetry boundary conditions were applied at the end nearest the indentation, and the y-displacement of the bottom edge of the neurite was prescribed as $e^{-5000z^2}$to approximate the displacement profile seen in Sanzeni et al. 2019. In the straight geometry, a z-displacement of -0.1um was applied to the end closer to the indenter to replicate strains from the other geometry. Y-displacement of the bottom edge in this case was fixed at zero. For all simulations, C3D8R

elements (8-node linear brick with reduced integration and hourglass control) were used, and a mesh convergence analysis was performed to find the appropriate element size.

Acknowledgements
Sherwood, Chalfie and Shen labs for sharing worm strains
Brunet lab for sharing RNAi bacterial cultures
CSIF for confocal equipment for FRAP experiments
SNF for patterning experiments
Zhiwen Liao for worm injections

Author contributions
Conceptualization, A.D., J.F., D.C., M.B.G.;
Methodology, A.D., J.F.;
Software, A.D., J.F., L.W., L.M.W.;
Validation, A.D., J.F., L.W.;
Formal analysis, A.D., J.F., L.W.;
Investigation, A.D., J.F., L.W., D.C.;
Resources, A.D., J.F., L.W., B.P., E.K., M.B.G.;
Data Curation, A.D., J.F., L.W.;
Writing - Original draft, A.D., J.F., L.W., D.C.;
Writing - Review and editing, A.D., J.F., L.W., D.C., M.B.G.;
Visualization, A.D., J.F., L.W., L.M.W.;
Supervision, A.D., M.B.G.;
Project administration, A.D., M.B.G.;
Funding acquisition, M.B.G.

Declaration of Interests
The authors declare no competing interests.

References

Axelrod, D., Koppel, D. E., Schlessinger, J., Elson, E., and Webb, W. W. (1976) Mobility measurement by analysis of fluorescence photobleaching recovery kinetics. Biophys. J. 16, 1055–1069

Akin, E.J., Solé, L., Dib-Hajj, S.D., Waxman, S.G., and Tamkun, M.M. (2015). Preferential targeting of Nav1.6 voltage-gated Na+ channels to the axon initial segment during development. PLoS One *10*.

Akin, E.J., Solé, L., Johnson, B., Beheiry, M. el, Masson, J.B., Krapf, D., and Tamkun, M.M. (2016). Single-Molecule Imaging of Nav1.6 on the Surface of Hippocampal Neurons Reveals Somatic Nanoclusters. Biophys. J. *111*, 1235–1247.

Akin, E.J., Higerd, G.P., Mis, M.A., Tanaka, B.S., Adi, T., Liu, S., Dib-Hajj, F.B., Waxman, S.G., and Dib-Hajj, S.D. (2019). Building sensory axons: Delivery and distribution of NaV1.7 channels and effects of inflammatory mediators. Sci. Adv. *5*, 4755.

Árnadóttir, J., O'Hagan, R., Chen, Y., Goodman, M.B., Chalfie, M., Arnadóttir, J., O'Hagan, R., Chen, Y., Goodman, M.B., Chalfie, M., et al. (2011). The DEG/ENaC protein MEC-10 regulates the transduction channel complex in Caenorhabditis elegans touch receptor neurons. J. Neurosci. *31*, 12695–12704.

Aumailley, M. (2013). The laminin family. Cell Adhes. Migr. *7*, 48–55.

Chelur, D.S., Ernstrom, G.G., Goodman, M.B., Yao, C.A., Chen, L., O'Hagan, R., Chalfie, M., O' Hagan, R., Chalfie, M., O'Hagan, R., et al. (2002). The mechanosensory protein MEC-6 is a subunit of the C. elegans touch-cell degenerin channel. Nature *420*, 669–673.

Chen, Y., Bharill, S., Isacoff, E.Y., and Chalfie, M. (2015). Subunit composition of a DEG/ENaC mechanosensory channel of Caenorhabditis elegans. Proc. Natl. Acad. Sci. U. S. A. *112*, 11690–11695.

Chiang, L.Y., Poole, K., Oliveira, B.E., Duarte, N., Sierra, Y.A.B., Bruckner-Tuderman, L., Koch, M., Hu, J., and Lewin, G.R. (2011). Laminin-332 coordinates mechanotransduction and growth cone bifurcation in sensory neurons. Nat. Neurosci. *14*, 993–1000.

Christensen, A.P., and Corey, D.P. (2007). TRP channels in mechanosensation: direct or indirect activation? Nat. Rev. Neurosci. *8*, 510–521.

Colognato, H., Winkelmann, D.A., and Yurchenco, P.D. (1999). Laminin Polymerization Induces a Receptor–Cytoskeleton Network Holly. *145*.

Cueva, J.G., Mulholland, A., and Goodman, M.B. (2007). Nanoscale organization of the MEC-4 DEG/ENaC sensory mechanotransduction channel in Caenorhabditis elegans touch receptor neurons. J. Neurosci. *27*, 14089–14098.

Du, H., Gu, G., William, C.M., and Chalfie, M. (1996). Extracellular proteins needed for C. elegans mechanosensation. Neuron *16*, 183–194.

Eastwood, A.L., and Goodman, M.B. (2012). Insight into DEG/ENaC channel gating from genetics and structure. Physiology *27*, 282–290.

Emtage, L., Gu, G., Hartwieg, E., and Chalfie, M. (2004). Extracellular proteins organize the mechanosensory channel complex in C. elegans touch receptor neurons. Neuron *44*, 795–807.

Gopal, S., Søgaard, P., Multhaupt, H.A.B., Pataki, C., Okina, E., Xian, X., Pedersen, M.E., Stevens, T., Griesbeck, O., Park, P.W., et al. (2015). Transmembrane proteoglycans control stretch-activated channels to set cytosolic calcium levels. J. Cell Biol. *210*, 1199–1211.

Gu, G., Caldwell, G.A., and Chalfie, M. (1996). Genetic interactions affecting touch sensitivity in *Caenorhabditis elegans*. Proc. Natl. Acad. Sci. U. S. A. *93*, 6577–6582.

Hayakawa, K., Tatsumi, H., and Sokabe, M. (2008). Actin stress fibers transmit and focus force to activate mechanosensitive channels. J. Cell Sci. *121*, 496–503.

Hu, J., Chiang, L.-Y., Koch, M., and Lewin, G.R. (2010). Evidence for a protein tether involved in somatic touch. EMBO J. *29*, 855–867.

Huang, C.C., Hall, D.H., Hedgecock, E.M., Kao, G., Karantza, V., Vogel, B.E., Hutter, H., Chisholm, A.D., Yurchenco, P.D., and Wadsworth, W.G. (2003). Laminin α subunits and their role in C. elegans development. Development *130*, 3343–3358.

Kao, G., Huang, C.-C.C., Hedgecock, E.M., Hall, D.H., and Wadsworth, W.G. (2006). The role of the laminin beta subunit in laminin heterotrimer assembly and basement membrane function and development in C. elegans. Dev. Biol. *290*, 211–219.

Katta, S., Krieg, M., and Goodman, M.B. (2015). Feeling Force: Physical and Physiological Principles Enabling Sensory Mechanotransduction. Annu. Rev. Cell Dev. Biol. *31*, 347–371.

Katta, S., Sanzeni, A., Das, A., Vergassola, M., and Goodman, M.B. (2019). Progressive recruitment of distal MEC-4 channels determines touch response strength in C. elegans. J. Gen. Physiol. *151*, 1213–1230.

Keeley, D.P., Hastie, E., Jayadev, R., Kelley, L.C., Chi, Q., Payne, S.G., Jeger, J.L., Hoffman, B.D., and Sherwood, D.R. (2020). Comprehensive Endogenous Tagging of Basement Membrane Components Reveals Dynamic Movement within the Matrix Scaffolding. Dev. Cell *54*, 60-74.e7.
Kim, S., and Wadsworth, W.G. (2000). Positioning of longitudinal nerves in C. elegans by nidogen. Science (80-.). *288*, 150–154.
Krieg, M., Dunn, A.R., and Goodman, M.B. (2015). Mechanical systems biology of C. elegans touch sensation. *37*.
Martinac, B. (2014). The ion channels to cytoskeleton connection as potential mechanism of mechanosensitivity. Biochim. Biophys. Acta - Biomembr. *1838*, 682–691.
O'Hagan, R., Chalfie, M., and Goodman, M.B. (2005). The MEC-4 DEG/ENaC channel of Caenorhabditis elegans touch receptor neurons transduces mechanical signals. Nat. Neurosci. *8*, 43–50.
Peters, B., Hartle, R., Krzesicki, R., Kroll, T., Perini, F., Balun, J., Goldstein, I., and Ruddon, R. (1985). The biosynthesis, processing, and secretion of laminin by human choriocarcinoma cells. J. Biol. Chem. *260*, 14732–14742.
Petzold, B.C., Park, S.J., Mazzochette, E.A., Goodman, M.B., and Pruitt, B.L. (2013). MEMS-based force-clamp analysis of the role of body stiffness in C. elegans touch sensation. Integr. Biol. (United Kingdom) *5*, 853–864.
Ranade, S.S., Syeda, R., and Patapoutian, A. (2015). Mechanically Activated Ion Channels. Neuron *87*, 1162–1179.
Senyürek, I., Kempf, W.E., Klein, G., Maurer, A., Kalbacher, H., Schäfer, L., Wanke, I., Christ, C., Stevanovic, S., Schaller, M., et al. (2014). Processing of Laminin α Chains Generates Peptides Involved in Wound Healing and Host Defense. J. Innate Immun. *6*, 467–484.
Shaw, L., Williams, R.L., and Hamill, K.J. (2020). CRISPR-Cas9-mediated labelling of the C-terminus of human laminin β1 leads to secretion inhibition. BMC Res. Notes *13*, 90.
Struntz, P., and Weiss, M. (2016). Multiplexed measurement of protein diffusion in Caenorhabditis elegans embryos with SPIM-FCS Related content Anomalous transport in the crowded world of biological cells. J. Phys. D Appl. Phys *49*, 44002.
Tamkun, M.M., O'Connell, K.M.S., and Rolig, A.S. (2007). A cytoskeletal-based perimeter fence selectively corrals a sub-population of cell surface Kv2.1 channels. J. Cell Sci. *120*, 2413–2423.
Vásquez, V., Krieg, M., Lockhead, D., and Goodman, M.B. (2014). Phospholipids that contain polyunsaturated fatty acids enhance neuronal cell mechanics and touch sensation. Cell Rep. *6*, 70–80.
Zhang, S., Arnadottir, J., Keller, C., Caldwell, G.A., Yao, C.A.A., and Chalfie, M. (2004). MEC-2 Is recruited to the putative mechanosensory complex in C. elegans touch receptor neurons through Its stomatin-like domain. Curr. Biol. *14*, 1888–1896.

Tables

Table 1: List of *C. elegans* strains used in this study

Strain name	Genotype	Source
Touch assay		
N2	*wild-type*	CGC
TU253	*mec-4(u253) X*	CGC
GN995	*mec-5(u444) X*	This study
CB1496	*mec-1(e1496) V*	Chalfie lab
CB3206	*mec-1(e1738) V*	Chalfie lab
CB1526	*mec-1 (e1526) V*	Chalfie lab
GN993	*mec-9(u437) V*	This study
TU34	*mec-9(u34) V*	Chalfie lab
CH119	*nid-1(cg119) V*	CGC
AH205	*sdn-1(zh20) X*	CGC
Puncta analysis		
GN753	*pgSi116[mec-17p::mNeonGreen::3XFLAG::mec-4::tbb-2 3'UTR] II*	This study
GN896	*lam-1(pg136[lam-1::wormScarlet::Lox2272::3xMyc]) IV*	This study
GN929	*nid-1(pg138[nid-1::wormScarlet::Lox2272::3xMyc]) V*	This study
GN956	*lam-3(pg140[lam-3::wormScarlet::Lox2272::3xMyc]) I*	This study
GN947	*epi-1(pg139[epi-1::wormScarlet::Lox2272::3xMyc]) IV*	This study
NK2335	*lam-2(qy20[lam-2::mNG+loxP]) X*	Sherwood lab
NK2425	*lam-3(qy28[lam-3::mNG+loxP]) I*	CGC
NK2404	*epi-1(qy31[epi-1::mNG+loxP]) IV*	CGC
NK2443	*nid-1(qy38[nid-1::mNG+loxP]) V*	CGC
GN851	*pgSi116 II; mec-5(u444) X*	This study
GN921	*pgSi116 II; mec-1(e1496) V*	This study
GN922	*pgSi116 II; mec-1(e1738) V*	This study
GN974	*pgSi116 II; mec-1(e1526) V*	This study
GN935	*pgSi116 II; mec-9(u437) V*	This study

GN975	*pgSi116 II; mec-9(u34) V*	This study
GN923	*pgSi116 II; nid-1(cg119) V*	This study
GN924	*pgSi116 II; dgn-1(cg121) X*	This study
GN927	*pgSi116 II; sdn-1(zh20) X*	This study
GN944	*pgSi116 II; fbl-1(hd43) IV*	This study
GN1003	*pgSi116 II; epi-1(gm57) IV*	This study
GN868	*pgSi116 II; unc-70(e524) V*	This study
GN852	*pgSi116 II; mec-7(ok2152) X*	This study
GN954	*pgSi116 II; pmk-3(ok169) IV; mec-7(ok2152) X*	This study
GN877	*pgSi116 II; pgSi129 I*	This study
GN878	*pgSi116 II; pgSi129 I; pat-2(ok2143) III*	This study
GN899	*pgSi116 II; pgSi129 I; him-4(e1267) X*	This study
GN915	*lam1(pg136) IV; mec-4(u253) X*	This study
GN916	*lam1(pg136) IV; mec-5(u444) X; uIs31 III*	This study
GN918	*lam1(pg136) IV; mec-1(e1496) V*	This study
GN933	*lam1(pg136) IV; mec-9(u437) V; uIs31 III*	This study
GN925	*lam1(pg136) IV; nid-1(cg119) V*	This study
GN920	*lam1(pg136) IV; dgn-1(cg121) X*	This study
GN926	*lam1(pg136) IV; sdn-1(zh20) X*	This study
GN965	*lam-2(qy20) X; uIs115*	This study
GN987	*lam-2(qy20)mec-4(u253) X; uIs115*	This study
GN982	*lam-2(qy20)mec-5(u444) X; uIs115*	This study
GN990	*lam-2(qy20) X; mec-1(e1496) V; uIs115*	This study
GN992	*lam-2(qy20) X; mec-1(e1738) V; uIs115*	This study
GN983	*lam-2(qy20) X; mec-1(e1526) V; uIs115*	This study
GN989	*lam-2(qy20) X; mec-9(u437) V; uIs115*	This study
GN988	*lam-2(qy20) X; mec-9(u34) V; uIs115*	This study
GN991	*lam-2(qy20) X; nid-1(cg119) V; uIs115*	This study
GN973	*nid-1(pg138) V; uIs31 III*	This study
GN971	*nid-1(pg138) V; mec-4(u253) X; uIs31 III*	This study

GN972	*nid-1(pg138) V; mec-5(u444) X; uIs31 III*	This study
GN1002	*nid-1(pg138)mec-1(e1496) V; uIs31 III*	This study
GN1001	*nid-1(pg138)mec-1(e1738) V; uIs31 III*	This study
GN996	*nid-1(pg138)mec-1(e1526) V; uIs31 III*	This study
GN997	*nid-1(pg141)mec-9(u437) V; uIs31 III*	This study
GN998	*nid-1(pg142)mec-9(u34) V; uIs31 III*	This study
Electrophysiology		
TU2769	*uIs31 III*	CGC
GN932	*uIs31 III; nid-1(cg119) V*	This study
TU2969	*uIs31 III; mec-5(u444) X*	Chalfie lab
GN985	*uIs31 III; mec-1(e1526) V*	This study
GN986	*uIs31 III; mec-9(u34) V*	This study
Colocalization		
GN902	*pgSi116 II; lam-1(pg136) IV*	This study
GN941	*pgSi116 II; nid-1(pg138) V*	This study
GN966	*pgSi116 II; pat-2(pg125) III*	This study
GN940	*lam-2(qy20) X; nid-1(pg138) V*	This study
GN939	*lam-2(qy20) X; lam-1(pg136) IV*	This study
GN1004	*pgSi116 II; nid-1(pg138)mec-1(e1526) V*	This study
GN1005	*lam-2(qy20) X; nid-1(pg138)mec-1(e1526) V*	This study
GN1000	*pgSi116 II; lam-1(pg136) IV; mec-1(e1526) V*	This study
GN999	*pgSi116 II; lam-1(pg136) IV; nid-1(cg119) V*	This study
TV2411	*wyIs97*	Shen Lab
GN981	*pgSi116 II; wyIs97 ?*	This study

Figure legends

Figure 1	Visual screen for extracellular matrix proteins associated with touch receptor neurons (TRNs) *in vivo* (top to bottom) Endogenous loci encoding LAM-2 ɣ lamnin, NID-1 nidogen, SDN-1 syndecan, FIB-1 fibulin, HIM-4 hemicentin, SAX-7 nrCAM, UNC-52 perlecan were tagged with mNeonGreen. Anterior is to the left, ventral to the bottom. Scale bar = 10µm.
Figure 2	NID-1, laminin, and MEC-4 colocalize to microscale puncta distributed along the length of TRN neurites. NID-1 colocalizes with NID-1 (A), LAM-2 ɣ lamnin (B), and MEC-4 (C). LAM-1 β lamnin colocalizes with LAM-2 ɣ laminin (D). MEC-4 does not colocalize with PAT-2 β integrin (E) but does colocalize with LAM-1 (F). MEC-4, LAM-2, and NID-1 are largely immobile. Fluorescence recovery after bleaching (FRAP) and time-lapse imaging show that puncta are immobile and do not recover from bleaching over a 5-minute observation period (G, H, I). Average time course of fluorescence recovery of 25 MEC-4 puncta (25 neurons) (J), 10 LAM-2 puncta (10 neurons) (K), and 11 NID-1 puncta (1 neurons) (L). Lines are the average of all FRAP experiments and the shaded area shows the standard deviation.
Figure 3	*mec-4, mec-5, mec-9,* and *mec-1* loss of function mutants are severely touch insensitive, whereas *nid-1* and *sdn-1* mutants have mild or no impairment (A). Mechanoreceptor currents (MRCs) are retained in *nid-1* mutants but lost in *mec-1* mutants (B). MRCs at onset of stimulation (C) and the offset of stimulation (D) are smaller in *nid-1* mutants compared to wild-type controls and absent in *mec-1* mutants.
Figure 4	*mec-1(e1526)* and *nid-1(cg119)* mutants retain MEC-4 puncta (A), but they are more widely separated in these mutants relative to wildtype (B). Isolated TRNs cultured on peanut lectin stripes have very few puncta (C) and they are widely separated (D).
Figure 5	Motile MEC-4 puncta are in RAB-3-expressing vesicles. Motile MEC-4 puncta are rare *in vivo* (A) and common in isolated TRNs (B). In both settings, motile MEC-4 puncta co-localize with RAB-3-expressing vesicles (C, D). The proportion of motile MEC-4 puncta is higher in isolated TRNs than in neurons in living animals. MEC-4 is tagged with mNG (*pSi116)*; RAB-3 is tagged with mCherry (*wyIs348*). Scale bar is 5 µm.
Figure 6	Laminin and Nidogen puncta distribution depend on other ECM proteins but not MEC-4. Laminin (A) and NID-1 (B) puncta decorate the TRNs. These puncta are independent of *mec-4* expression (second row) and lost in *mec-1, mec-5*, and *mec-9* loss-of-function mutants. The *mec-1(e1526)* allele retains these puncta as well as MEC-4 puncta (see Figure 4).
Figure 7	(in progress) Laminin/MEC-5/MEC-9 puncta distribution in nidogen null. Nidogen/MEC-5/MEC-9 puncta distribution in laminin RNAi.
Figure 8	Force from filament: ECM proteins focus mechanical strain at MEC-4 channels

Figure 1

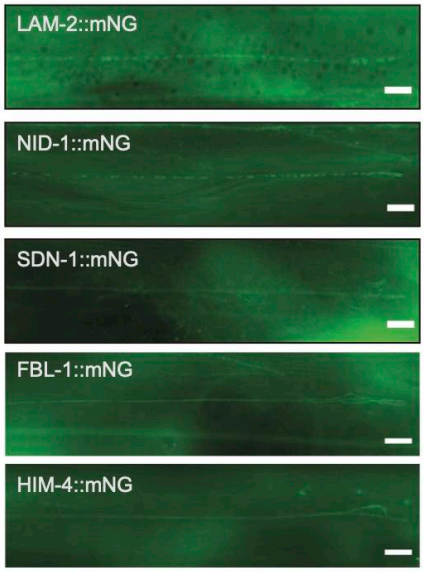


Figure 2

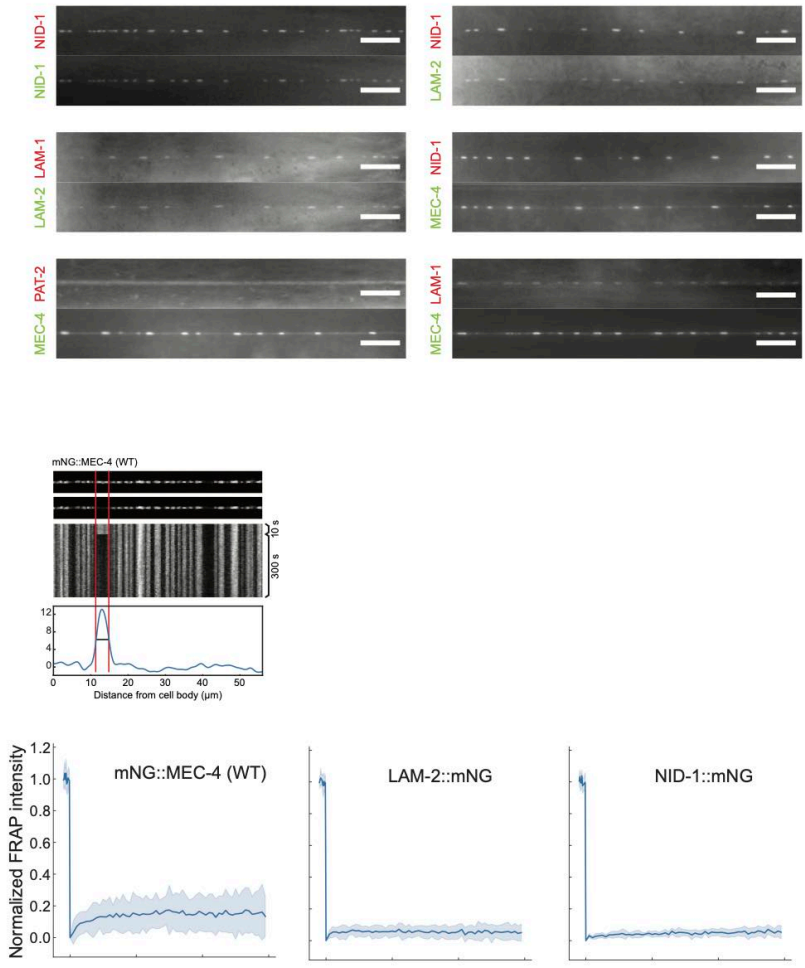
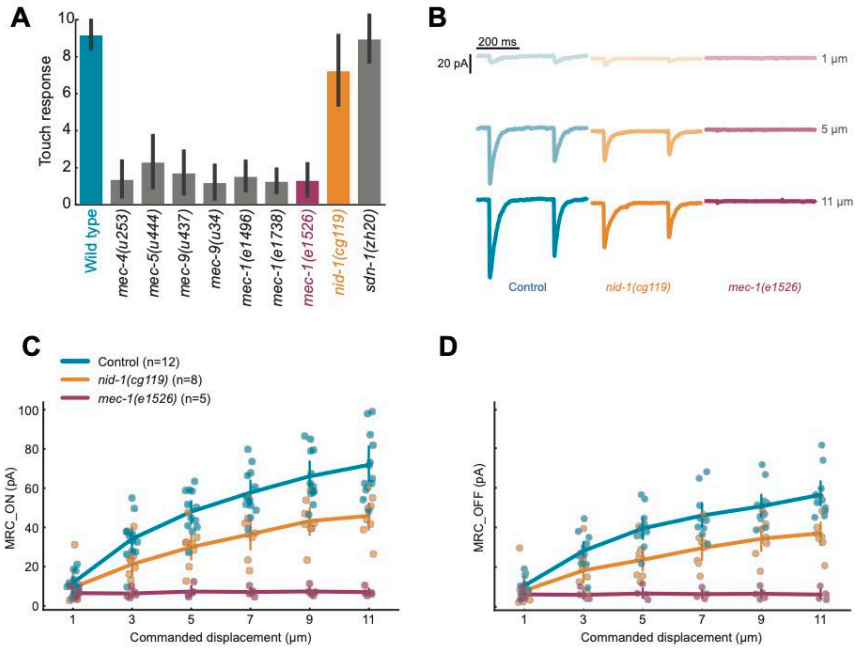

NID-1
NID-1
NID-1
LAM-2
LAM-1
LAM-2
NID-1
MEC-4
PAT-2
MEC-4
LAM-1
MEC-4
mNG::MEC-4 (WT)
10 s
300 s
12
8
4
0
0
10
20
30
40
50
Distance from cell body (µm)
Normalized FRAP intensity
1.2
1.0
0.8
0.6
0.4
0.2
0.0
mNG::MEC-4 (WT)
LAM-2::mNG
NID-1::mNG

Figure 3


A
Touch response
10
8
6
4
2
0
Wild type
mec-4(u253)
mec-5(u444)
mec-9(u437)
mec-9(u34)
mec-1(e1496)
mec-1(e1738)
mec-1(e1526)
nid-1(cg119)
sdn-1(zh20)
B
200 ms
20 pA
1 µm
5 µm
11 µm
Control
nid-1(cg119)
mec-1(e1526)
C
Control (n=12)
nid-1(cg119) (n=8)
mec-1(e1526) (n=5)
MRC_ON (pA)
100
80
60
40
20
0
1
3
5
7
9
11
Commanded displacement (µm)
D
MRC_OFF (pA)
1
3
5
7
9
11
Commanded displacement (µm)

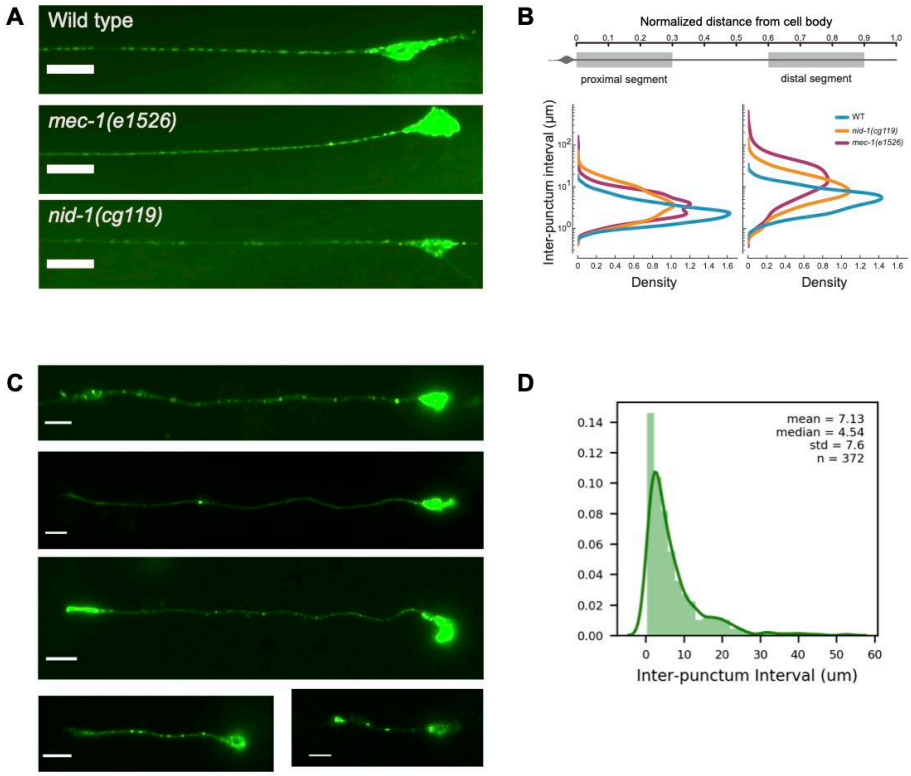

A
Wild type
mec-1(e1526)
nid-1(cg119)
B
Normalized distance from cell body
proximal segment
distal segment
Inter-punctum interval (µm)
WT
nid-1(cg119)
mec-1(e1526)
Density
Density
C
D
mean = 7.13
median = 4.54
std = 7.6
n = 372
Inter-punctum Interval (um)

FIGURE 4

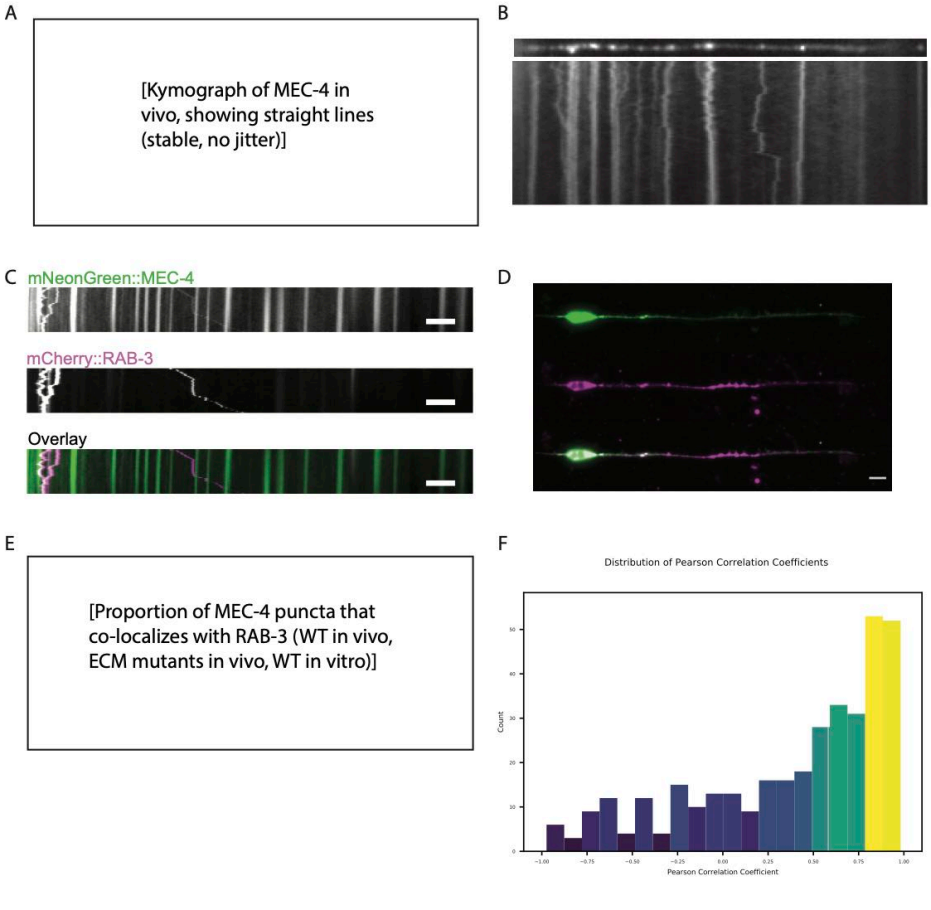

A
[Kymograph of MEC-4 in vivo, showing straight lines (stable, no jitter)]
B
C
mNeonGreen::MEC-4
mCherry::RAB-3
Overlay
D
E
[Proportion of MEC-4 puncta that co-localizes with RAB-3 (WT in vivo, ECM mutants in vivo, WT in vitro)]
F
Distribution of Pearson Correlation Coefficients
Count
Pearson Correlation Coefficient

FIGURE 5

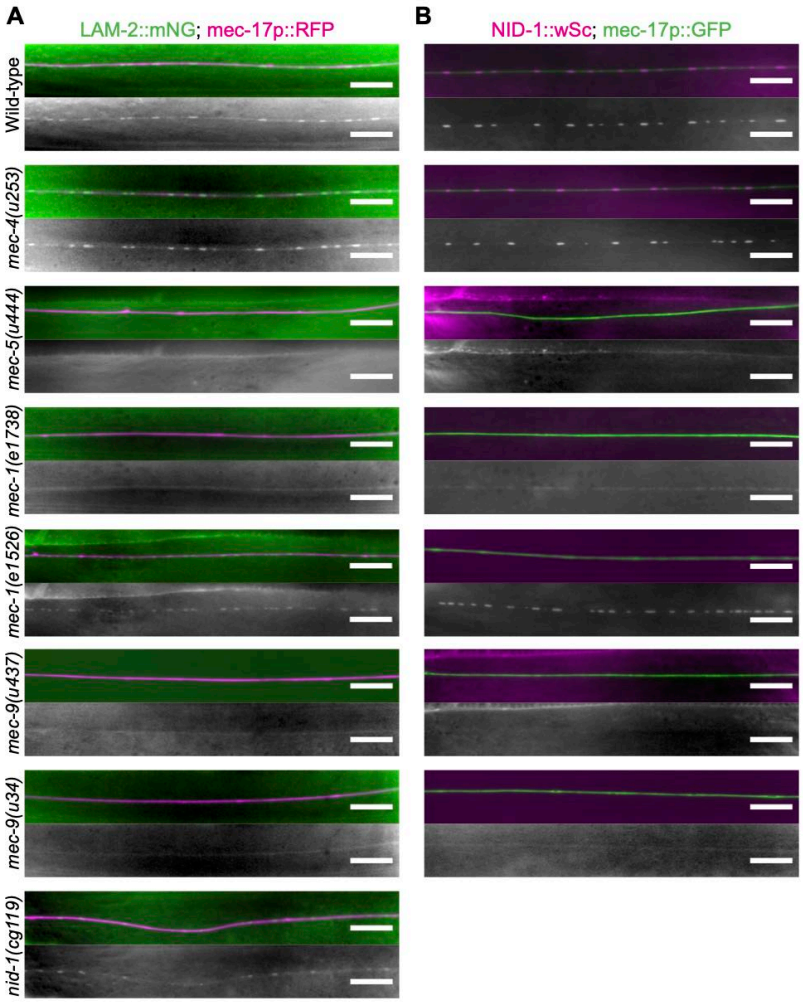

A
LAM-2::mNG; mec-17p::RFP
B
NID-1::wSc; mec-17p::GFP
Wild-type
mec-4(u253)
mec-5(u444)
mec-1(e1738)
mec-1(e1526)
mec-9(u437)
mec-9(u34)
nid-1(cg119)

FIGURE 6

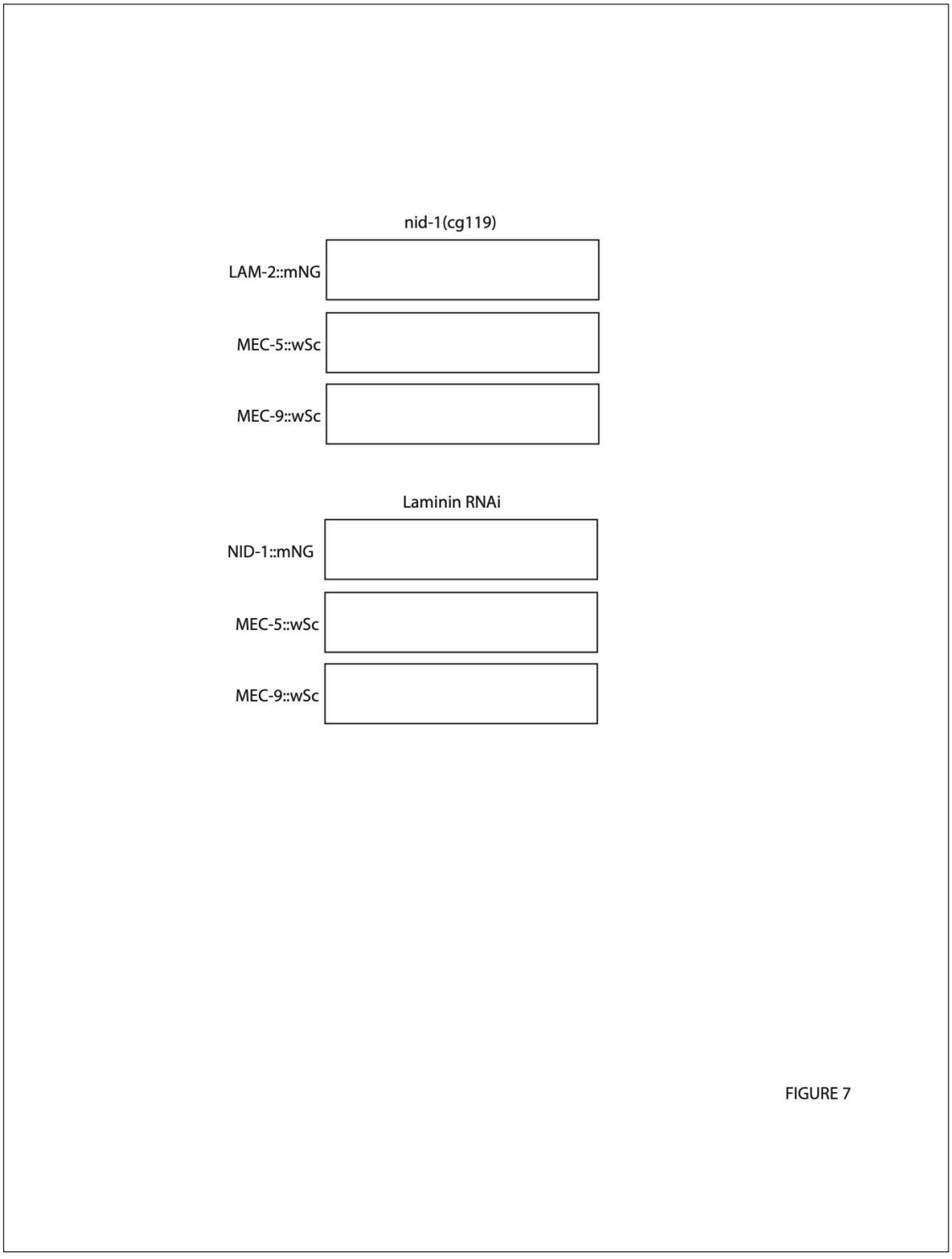

nid-1(cg119)
LAM-2::mNG
MEC-5::wSc
MEC-9::wSc
Laminin RNAi
NID-1::mNG
MEC-5::wSc
MEC-9::wSc

FIGURE 7

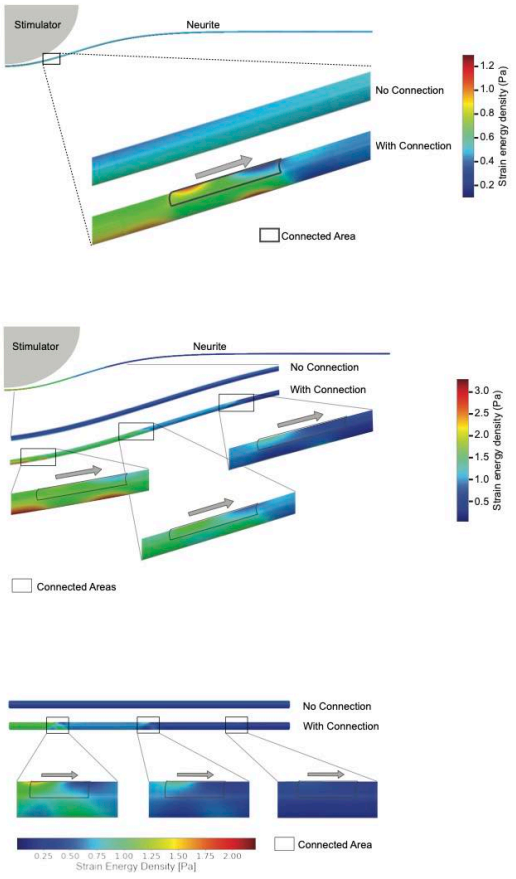

Stimulator
Neurite
No Connection
With Connection
Connected Area
1.2
1.0
0.8
0.6
0.4
0.2
Strain energy density (Pa)
Stimulator
Neurite
No Connection
With Connection
Connected Areas
3.0
2.5
2.0
1.5
1.0
0.5
Strain energy density (Pa)
No Connection
With Connection
Connected Area
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
Strain Energy Density [Pa]

FIGURE 8

Chapter 5

Conclusion

5.1 Summary

Cell culture systems play a vital role in biological research by providing experimenters with reduced order platforms for testing hypotheses where results may be confounded by competing factors in vivo. In the case of investigating form-function coupling, redirecting neurite outgrowth would alter morphology but it would also alter the neurite's immediate environment. By isolating and culturing the mechanoreceptor neurons in a controlled system, the shape can be controlled with a greater degree of independence from other experiment parameters. However, these cell culture systems suffer from the common problems of variation and reproducibility. Unlike engineering materials, biological specimens often show greater variability in their physical properties such as morphology and mechanics. The widespread distribution of values about the population mean makes statistical testing of hypotheses rather difficult unless the experimental perturbation results in a large effect, the sample sizes are large, or both. The greater deviation of measured values from the mean correlates with a lower signal-to-noise ratio which reduces the experimental certainty. This is something experimenters would be wise to consider when choosing a system for investigating a biological process. Although *C. elegans* themselves are considered among some of the most stereotyped and genetically uniform, their embryonic cell cultures are extremely heterogeneous, containing both post-mitotic and dividing cells and a large number of differentiated classes

of cell types. This heterogeneity is likely one of the reasons the *in vitro* approach is underutilized.

To overcome substantial limitations to the accessibility of *in vitro* methods, this thesis presents a multi-faceted approach to reducing variability.

- I developed genetic, micropatterning, image analysis tools for minimizing cultured TRN population variance and measurement observation space. Although this work has focused on the touch receptor neurons, extending these experimental strategies to other classes of neurons is expected to be straightforward. Experimenters wanting to work with isolated cells typically choose from two options: 1)An immortalized cell line of a single type (e.g., muscle, skin, neuron, or cancer-like cells); 2)Primary cells from dissociated tissue. Since the former does not currently exist for *C. elegans*, the only appropriate choice for studying form-function relationships *in vitro* for *C. elegans* cells is primary culture of cells isolated from dissociated embryos. As the embryos are generated asynchronously, the resulting cells are highly heterogeneous both in cell type and differentiation state.

- I investigated the feasibility of using transgenic antibiotic resistance to enrich the cultures for particular cell types. To do this I leveraged an existing flexible vector for CRISPR/Cas9 single copy insertion of the puromycin resistance (puroR) gene.

- I developed three transgenes that expressed the puroR gene under a TRN, hypodermal, or pan-neuronal promoter. The synthetic puroR gene used the SL2 splice leader to express a fluorescent reporter from the same promoter. This approach enabled the visualization of transgene expression.

- Previous studies had established micropatterning of proteins as a reliable method for guiding cell shape *in vitro* [106], though no group had applied this approach to *C. elegans* cells. To test if I could use micropatterning to control TRN morphology, I used a contactless, photolithography system to biofabricate geometrically defined regions of the peanut agglutinin (PNA) surrounded by a bio-passivating layer of polyethylene-glycol (PEG). I found that if the feature size was sufficiently small, a majority of the cultured TRNs would extend neurites along the length of the micropattern. These narrow lines of PNA surrounded by PEG

permitted TRNs to adopt their usual unipolar morphology and seemd to suppress secondary branching commonly observed in cultures without micropatterns.

- I further refined this patterning strategy to allow for the addition of a second protein, EHS-laminin, to test if this exogenous protein could improve the organization of the molecular-scale mechanosensing machinery within the TRN.

- Although the TRN is the mechanoreceptor neuron responsible for detecting low-threshold mechanical stimuli in *C. elegans*, the fundamental sensing unit is transmembrane, pore-forming protein MEC-4. This protein forms ion channels in the TRN's plasma membrane that open and close in response to the application or removal of mechanical stress. Prior to this thesis, whether or not the isolated TRNs would express a functional version of this protein remained unknown. Preliminary evidence from my bioAFM experiments had suggested that a subpopulation of cultured mechanoreceptors may indeed possess MEC-4 ion channels capable of detecting mechanical stimuli. Again, the biological variability between specimens posed a serious challenge for experimental work and so I tested if micropatterning could be used to create cultures of TRNs more reliably expressing these ion channels. To do this, I utilized a transgenic animal developed by others expressing a version of MEC-4 protein containing an mNeonGreen fluorophore fused to the N-terminus of the protein. This molecular tool allowed me to visualize the protein in the isolated cells.

- To quantify this localization of MEC-4 in cells on micropatterns, I developed a suite of image analysis tools used in an automated or supervised mode to measure neurite length and MEC-4 distribution. I further developed a set of code that uses machine learning to develop a statistical model of what aggregations of signal coming from the MEC-4 protein could be classified as bonafide "puncta" that are more likely to be assembled ion channels rather than individual proteins.

- Through this work I found that isolated TRNs are significantly shorter and possess fewer puncta per unit length than their *in vivo* counterparts in both the presence and absence of a second cell adhesion molecule.

5.2 Limitations

Although isolated TRNs from *C. elegans* show promise for form-function and other mechanobiology studies, the results of this thesis are met with certain limitations that warrant consideration. Cells from *C. elegans* embryos are substantially smaller than those from mammalian systems. TRN cell bodies are no bigger than 5 μm with neurites less than 500 nm in diameter. This small size manifests as dim cells that make it difficult to delineate bona fide cells from debris in flow cytometry analysis. This complicated studies aimed at quantifying puromycin efficacy using flow cytometry. I was also confronted by confounding results with cells in culture appearing to withstand unusually high levels of puromycin, as suggested by the use of commercial live-dead staining kits. It may have been that this marker was not optimized for invertebrate cells, which is an additional limitation of working with this system. The majority of commercially available cell culture regents are geared towards mammalian cells and researchers may need to invest effort into optimizing materials for C. elegans studies.

Smaller cells also means that any fabricated device will need to be scaled down to present appropriate physical boundaries. In the case of micropatterns, the smallest feature size achievable with the LIMAP-based PRIMO instrument (Alvéole) is 3 μm As this is much wider than the TRN neurite, this thesis did not explore how smaller feature sizes might alter neurite morphology. As a practical matter, small cells complicates tasks such as quantifying cell seeding density with a hematocytometer. With regards to additional mechanobiology assays, potential experimenters will also need to consider how cell-substrate adhesion for isolated cells from *C. elegans* may impact their findings. For the experiments of this thesis as well as those from prior to this work, PNA is a the only known molecule that supports attachment to glass substrates. Based on my preliminary analysis, this suggests that the cells are attaching to substrates using carbohydrate-lectin binding, which has uncharacterized tensile strength. Thus if experiments involve applying mechanical stress to the samples, one will need to factor in the unknown mechanics of this system. For example, in assays where cells are cultured on a stretchable substrate and then subjected to tensile stress, the cells may detach from the substrate.

The overarching theory applied by this thesis is that morphological measurements in low signal-to-noise systems, such as a heterogeneous cell culture system, can be improved by applying engineering principles. The techniques used to demonstrate this theory have focused on reducing population heterogeneity, morphological variability, and the dependencies on operator judgement. Together, these measures reduce variance, improve robustness, rigor, and reproducibility of these studies of neuronal cell biology. Nonetheless, they are not without limitations. In each of these approaches there remain potential sources of bias that can contribute to the underlying measurement uncertainty. By reducing population heterogeneity with puromycin-based enrichment stategies there may be side effects of caused by the antibiotic that might limit the utility of the cell culture system (e.g., uncharacterized effects on organelle production). In the case of restricting the observation space by only analyzing TRNs with minimum length requirements, this may introduce a source of selection bias. These are natural trade-offs that cannot be avoided when attempting to standardize a measurement process, which is all the more reason why they warrant attention in future experiments.

5.3 Future Directions

5.3.1 Studying TRNs *in vitro*

This thesis generated several unexpected findings. At the start of this work it unknown whether or not MEC-4 would be present along the TRN neurite *in vitro*. All prior work had suggested we would not. We were surprised to see that these MEC-4 puncta that are stable and stationary *in vivo* were mobile in cultured TRNs. This finding was only observed because we cultured cells expressing the tagged version of MEC-4 and did time-lapse imaging of live cells over a 16 hour period. It was in these very preliminary experiments that we saw a dramatic change in the position of the only puncta along the neurite. From this finding we hypothesized that the observed puncta would not be positionally stable without the ECM, based on existing force-from-filament models of MeT channel gating that suggest the MeT channel is tethered to the ECM. We were motivated to pursue this research direction, and completed an additional studies that showed the majority of the MEC-4 puncta colocalized with a known vesicle marker (these finding are detailed in Chapter 4).

From these results we inferred that the colocalized MEC-4 proteins might be undergoing active trafficking. Future work that could overcome the limitations of phototoxicity would yeild immediate benefits, including the ability to quantify motion and to test if mobility is dependent on molecular motors known to be expressed in the TRNs.

This small finding has challenged our existing paradrigm of how MeT channel distribution is maintained along the neurite. A handful of studies from others [60, 115] had previously used MEC-4 antibodies to estimate expression of MEC-4, posing the idea that MeT channel expression level could change during the adult animal's life. However, these studies did not reveal the details of how the protein would move from the soma to the distal regions of the neurite. Together with our findings, there is now more evidence in support of a model in which MeT proteins are trafficked in synaptic-like vesicles and this trafficking function is cell-autonomous. This nuanced point had previously been overlooked simply because (1) mutations to ECM encoding genes eliminated MEC-4 puncta, and (2) only two papers had published data on MEC-4 expression patterns *in vitro*. With this new understanding future studies, using optimized time-lapse imaging methods, should further dissect the molecular mechanisms involved in positioning MEC-4 puncta along the neurite. This would be an exciting finding that could enable the eventual reconstitution of TRN mechanosensitivity *in vitro*.

5.3.2 Extension to other applications

One of the great advantages of this cell culture system is the use of these cells for cryoET studies. This application was briefly discussed in Chapter 2, with preliminary data shown in Figure 2.2. Here, the small size of *C. elegans* neurons reduces experiment complexity since it eliminates the need for cryoFIB-milling samples for compatibility with cryo-electron microscopy. Others have demonstrated that the micropatterning techniques described in Chapter 3 can easily be extended to creating similar patterns on cryoET grids [109]. In this thesis, I chose to analyze cultured TRNs on micropatterns to reduce the morphological heterogeneity that made robust image analysis approaches intractable. Furthermore, I asserted that changing TRN morphology would alter microtubule organization. This is a reasonable hypothesis based on what others have shown, but this is also an excellent entry point for a detailed analysis of any accompanying changes in the nanoscale

architecture of the axonal cytoskeleton. Future studies should use published resources to adapt the micropatterning protocol in Chapter 3 for cryoET grids and analyze subcellular organization. Although implemented for TRNs, this approach is easily extended to other cell types. TRNs are among the rarest of cell types present, making up only about 1% of the total cell population. The cultures have many other neuron types, hypodermal cells, and muscle to name a few. These cell types are highly amenable to micropatterning assays.

Large scale image analysis continues to be a bottle neck in biological research. Image acquisition is a time consuming process that can generate data that is challenging to analyze and quantify in a robust manner. Manual analysis is common, and often adopted with consideration of the error-prone and tedious nature. This is rapidly evolving as more research groups adopt machine learning approaches to image analysis. The generation of a rigorous and unbiased training data and useful ground truth is not a trivial process, however. By taking an engineering design approach to addressing the need for methods enabling high-content generation and high-throughput analysis, I developed an optimized platform for sample preparation, data acquisition, analysis, and curation. All of these engineering methodologies are equally suitable for studies in other *C. elegans* cell types or other model systems (e.g., *Drosophila melanogaster*, HEK cells, primary mammalian neurons, or other cell types). The motivation for incorporating these approaches is to improve rigor, robustness, and experimental reproducibility. Doing so also holds the promise for lowering the measurement noise floor that can make it difficult to detect subtle changes in phenotype. Once such application that would be interesting to see in use is taking the methods developed in Chapter 2 and using them for neurotoxicity screens in response to various compounds.

The areas for future research described here broadly fall under the general theme of this thesis that is the improvement of biological measurements through the application of engineering principles. Although this theory has likely existed since the inception of scientific research, but the urgency of its application was driven home by the biomechanics pioneer YC Fung. In the early days of testing the mechanical properties of biological specimens Fung recognized that the measured properties were highly dependent on sample preparation and testing modality. This heterogeneity in measurement approaches made comparisons of material values from different groups nearly impossible. As the importance of mechanics in biological system becomes more and more clear, the

need for standardization of testing methods becomes even more apparent. This thesis applies this theory specifically to studying form function relationships in *C. elegans* touch receptor neurons, but the framework shown here is equally applicable to other areas of biology. Future research should absolutely consider how including engineering design methodologies into the iterative nature of biological research might serve to accelerate both fields

Appendix A

R-based Code for automatic classification of flow-analyzed live/dead and GFP(+/-) cells from *C. elegans* embryonic dissociations

The code used to classify live vs dead cells, as well as GFP(+/-) is broken down into three modules, where each module performs a specific function. The first module generates a file that stores all variables that is called and used by subsequent modules. For this reason, the first module must always be run prior to running the second or third. All code is written in the language R using open source packages that are freely available and are described in the comments of the markdown files. The code and files presented here are those used to generate the analysis presented in Chapter 2.

A.1 Classifier development from training data sets

This module uses ground truth data sets to determine cutoff values for cell size, live/dead, and GFP(+/-).

FCA_DisFig_Ch2_p1_Final

JFranco

4/13/2021

FCA Dissertation Figures for Ch2, part 1 - FINAL version

Part 1 of the analysis code uses experimental training data sets to determine live/dead and GFP(+/-) cutoffs for use in further classification. All generated values are saved as an R data file which is then reloaded in parts 2 and 3 for classifying experimental data sets.

Install the necessary packages

This installation procedure only needs to be conducted once. The code is left commented out as a helpful reminder if this is your first time setting up the R environment for FlowCytoAnalysis

```
#The code below should only need to be run once to install the necessary packages
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")

#BiocManager::install("flowCore")
#BiocManager::install("ggcyto")
#BiocManager::install("flowStats")
#install.packages("tidyverse")
#install.packages("scales")
#install.packages("ggpubr")
```

Load libraries

Although you do not need to install packages every time you run this code, you do need to load the libraries from said packages.

```
library(flowCore)
library(flowStats)
library(ggcyto)
library(ggplot2)
library(dplyr)
library(ggpubr)
require(scales)
require(gridExtra)
```

Specify the directory, folders, and file names for analysis

```
# Specified parameters
sbL.um = 4                # Lower bound for acceptable size of cells (um)
sbU.um = 10                # Upper bound for acceptable size of cells (um)

# Directories
dirBase <- "/Users/nerdette/Google Drive/"
dirData <- paste(dirBase,"Research/WormSense/Data/CCPs/",sep = '')
dirSave <- paste(dirBase,"Dissertation/2_AutoClass/Figures/Ch2_FCA/Final_4to10/",sep = '')

# Saving fsch size information
fnSCSV <- paste(dirBase, "Code/FlowCyto/FCA_Calibration/Calibration_Size.csv",
                sep='')

# Saving compensation information
fnComp <- paste(dirBase, "Code/FlowCyto/FCA_Calibration/Calibration_Comp.csv",
                sep='')

# Saving variables for P1
fnRdata <- "FCA_DissFig_Ch2_p1_Final_4to10.Rdata"

# Training data sets for size calibration
tdsSB <- c("CCP_068/Beads Files/Beads_1um.fcs",
           "CCP_068/Beads Files/Beads_2um.fcs",
           "CCP_068/Beads Files/Beads_4um.fcs",
           "CCP_068/Beads Files/Beads_6um.fcs",
           "CCP_069/Beads Files/Beads_1um.fcs",
           "CCP_069/Beads Files/Beads_2um.fcs",
           "CCP_069/Beads Files/Beads_4um.fcs",
           "CCP_069/Beads Files/Beads_6um.fcs",
           "CCP_070/Beads Files/Beads_1um.fcs",
           "CCP_070/Beads Files/Beads_2um.fcs",
           "CCP_070/Beads Files/Beads_4um.fcs",
           "CCP_070/Beads Files/Beads_6um.fcs")
j=1
for(i in tdsSB){
  tdsSB[j] <- paste(dirData,tdsSB[j],sep='')
  j = j + 1
}

# Training data sets for fluorescence
tdsN2.US <- "CCP_051/N2"              # Wild type cells for auto-fluorescence
tdsGFP.US <- "CCP_051/OH441"          # Mostly GFP(+) cells for FL1
# Reactive stained beads (bright red)
tdsPSB <- "CCP_070/Beads Files/Beads_Pos"
# Non-reactive stained beads (dim red)
tdsNSB <- "CCP_070/Beads Files/Beads_NSt"
tdsUSB <- "CCP_071/Beads Files/Beads_Neg"      # Unstained beads
tdsGFP.SL <- "CCP_078/08_OH441__0.0ugml-1"#GFP(+), stained, no lethal challenge
tdsGFP.PR <- "CCP_078/12_OH441__100ugml-1"    #GFP(+), stained, heat treated
tdsGFP.SD <- "CCP_078/07_OH441__HKC"    #GFP(+), stained, heat treated
tdsOFP.US <- "CCP_047/GN749"
```

```
# Data to be analyzed
# CTRL samples received no lethal challenge and were labeled with LIVE/DEAD
# marker
dsN2.S <- c("CCP_055/N2 Ctrl.fcs","CCP_056/N2 Ctrl.fcs")
j=1
for(i in dsN2.S){
  dsN2.S[j] <- paste(dirData,dsN2.S[j],sep='')
  j = j + 1
}
# Ethanol samples were incubated with ethanol for 30 minutes prior to labeling
# with the same LIVE/DEAD marker
dsN2.Eth.S <- c("CCP_055/N2 Ethanol.fcs","CCP_056/N2 Ethanol.fcs")
j=1
for(i in dsN2.Eth.S){
  dsN2.Eth.S[j] <- paste(dirData,dsN2.Eth.S[j],sep='')
  j = j + 1
}
# Other samples underwent either incubation with 100 ugml puromycin or were
# heat treated prior to labeling. Labeled with propidium iodide.
dsN2.Oth.S <- c("CCP_083/N2 100ugml.fcs", "CCP_083/10_N2_____HKC.fcs")
j=1
for(i in dsN2.Oth.S){
  dsN2.Oth.S[j] <- paste(dirData,dsN2.Oth.S[j],sep='')
  j = j + 1
}
```

Build flowsets and flowframes

```
ffN2.US <- read.FCS(paste(dirData,tdsN2.US,".fcs",sep=""), transformation=FALSE)
ffGFP.US <- read.FCS(paste(dirData,tdsGFP.US,".fcs",sep=""), transformation=FALSE)
ffPSB <- read.FCS(paste(dirData,tdsPSB,".fcs",sep=""), transformation=FALSE)
ffNSB <- read.FCS(paste(dirData,tdsNSB,".fcs",sep=""), transformation=FALSE)
ffUSB <- read.FCS(paste(dirData,tdsUSB,".fcs",sep=""), transformation=FALSE)
ffGFP.SL <- read.FCS(paste(dirData,tdsGFP.SL,".fcs",sep=""),
                     transformation=FALSE)
ffGFP.SP <- read.FCS(paste(dirData,tdsGFP.PR,".fcs",sep=""),
                     transformation=FALSE)
ffGFP.SD <- read.FCS(paste(dirData,tdsGFP.SD,".fcs",sep=""),
                     transformation=FALSE)
ffOFP.US <- read.FCS(paste(dirData,tdsOFP.US,".fcs",sep=""),
                     transformation=FALSE)

fsSB <- read.flowSet(tdsSB, transformation=FALSE)
fsN2.S <- read.flowSet(dsN2.S, transformation=FALSE)
fsN2.Eth.S <- read.flowSet(dsN2.Eth.S, transformation=FALSE)
fsN2.Oth.S <- read.flowSet(dsN2.Oth.S, transformation=FALSE)
```

Calibration of size<-> FSC-H relationship

Develop linear model to estimate size bounds

To estimate what bounds to use as cutoffs for gating cells based on size I will use a linear regression of data from the calibration beads. This is best done by transforming the flowset data into a dataframe

```
# Copy the values into a data frame
medians.fsch <- list()
j = 1
for(i in tdsSB){
medians.fsch[j] <- median(exprs(fsSB[[j]])[,"FSC-H"])
j = j+1
}
# Set up the rest of the values in the data frame
j=1
for(i in tdsSB){
  pData(fsSB)$prep[j] <- substr(tdsSB[j],75,77)
  pData(fsSB)$size[j] <- as.numeric( as.character( substr(tdsSB[j],85,85) ) )
  pData(fsSB)$median.fsch[j] <- as.numeric( as.character(medians.fsch[j]) )
  j= j+1
}
dfSB <- pData(fsSB)
# Generate the linear model and get the coefficients for making the prediction
lmSB = lm( formula = dfSB$median.fsch ~ dfSB$size)
print("Summary data for FSC-H ~ Bead Size Linear Regression")
```

```
## [1] "Summary data for FSC-H ~ Bead Size Linear Regression"
```

```
summary(lmSB)
```

```
##
## Call:
## lm(formula = dfSB$median.fsch ~ dfSB$size)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -86345 -38442  13009  48843  60297
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -99866      34736  -2.875   0.0165 *
## dfSB$size     136360       9202  14.819 3.93e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61210 on 10 degrees of freedom
## Multiple R-squared:  0.9564, Adjusted R-squared:  0.9521
## F-statistic: 219.6 on 1 and 10 DF,  p-value: 3.929e-08
```

```
cf.fsch <- coefficients(lmSB)
sbL = as.numeric( signif( cf.fsch[1] + cf.fsch[2]*sbL.um , digits = 4) )
sbU = as.numeric( signif( cf.fsch[1] + cf.fsch[2]*sbU.um, digits = 4) )
print( paste( "Lower bound of fsch (",sbL.um,"um) = ",  sbL))
```

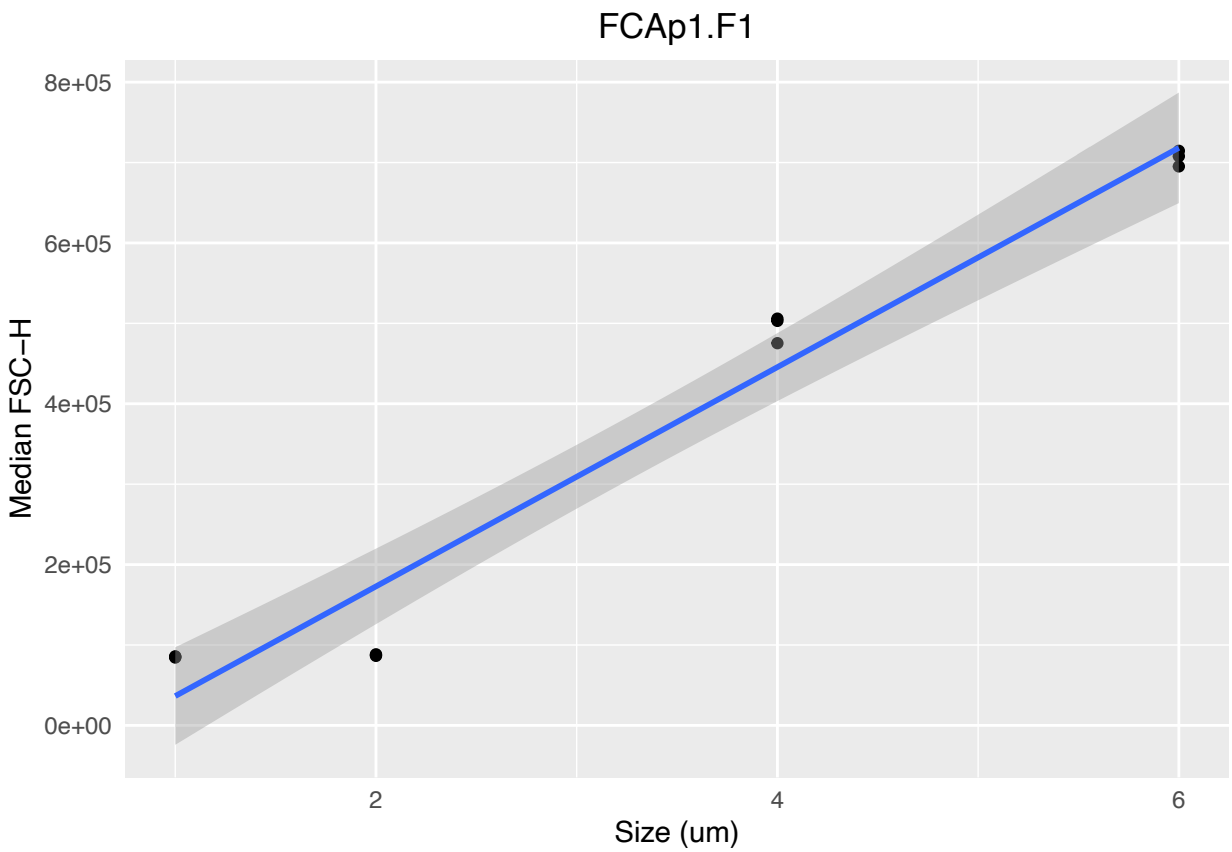
```
## [1] "Lower bound of fsch ( 4 um) =  445600"
```

```
print( paste( "Upper bound of fsch (",sbU.um,"um) = ",  sbU))
```

```
## [1] "Upper bound of fsch ( 10 um) =  1264000"
```

Plot FSCH for calibration beads

```
t <- "FCAp1.F1"
p <- ggplot(dfSB, aes(x=size, y=median.fsch)) +
  geom_point() +
  geom_smooth(method=lm) +
  xlab("Size (um)") +
  ylab("Median FSC-H") +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())+
  ggtitle(t)
p
```




```
ggsave(paste(dirSave,"SizeBeadsLM.png",sep=""), p, dpi = 300)
```

Create a data frame for storing values and writing to CSV

```
r1 <- data.frame("Bound"="Lower","Size in um"=sbL.um,"FSC-H"= sbL)
r2 <- data.frame("Bound"="Upper","Size in um"=sbU.um,"FSC-H"= sbU)

dfBounds <- rbind(r1,r2)
```

Write calculated bounds to file

```
write.csv(dfBounds, file = fnSCSV, row.names = FALSE)
```

Generate the gating object and subset the data

Values calculated in step 2 are used to remove any events that are too small or too big to likely be a cell.

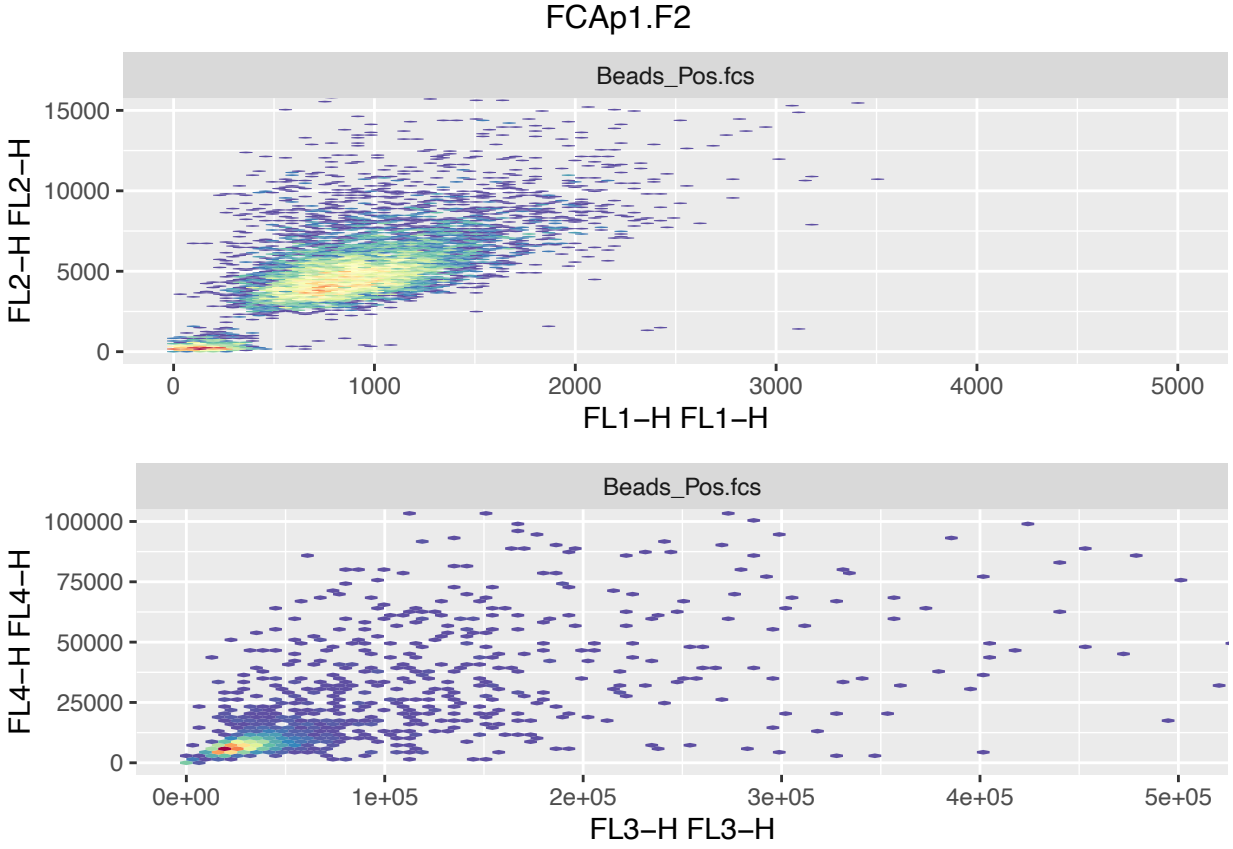
```
# Create gating objects
gtSize <- rectangleGate(filterId = "FSCH Size Gate",
                                    "FSC-H"=c(sbL,sbU))
ffN2.US.1 <- Subset(ffN2.US, gtSize)
ffGFP.US.1 <- Subset(ffGFP.US, gtSize)
ffPSB.1 <- Subset(ffPSB, gtSize)
ffNSB.1 <- Subset(ffNSB, gtSize)
ffUSB.1 <- Subset(ffUSB, gtSize)
ffGFP.SL.1 <- Subset(ffGFP.SL, gtSize)
ffGFP.SP.1 <- Subset(ffGFP.SP, gtSize)
ffGFP.SD.1 <- Subset(ffGFP.SD, gtSize)
ffOFP.US.1 <- Subset(ffOFP.US, gtSize)

fsN2.S.1 <- Subset(fsN2.S, gtSize)
fsN2.Eth.S.1 <-Subset(fsN2.Eth.S, gtSize)
fsN2.Oth.S.1 <-Subset(fsN2.Oth.S, gtSize)
```

Visualize ground truth data sets

Stained, Reactive beads (Beads_Pos)

These beads are coated in amine groups and have been stained

```
t <- "FCAp1.F2"
p1 <- ggcyto(ffPSB,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 256) +
  coord_cartesian(xlim = c(0, 5e3), ylim = c(0, 15e3))
p2 <- ggcyto(ffPSB,aes(x='FL3-H',y='FL4-H')) +
  geom_hex(bins = 256) +
  coord_cartesian(xlim = c(0, 5e5), ylim = c(0, 1e5))
grid.arrange(as.ggplot(p1),as.ggplot(p2),top=t)
```



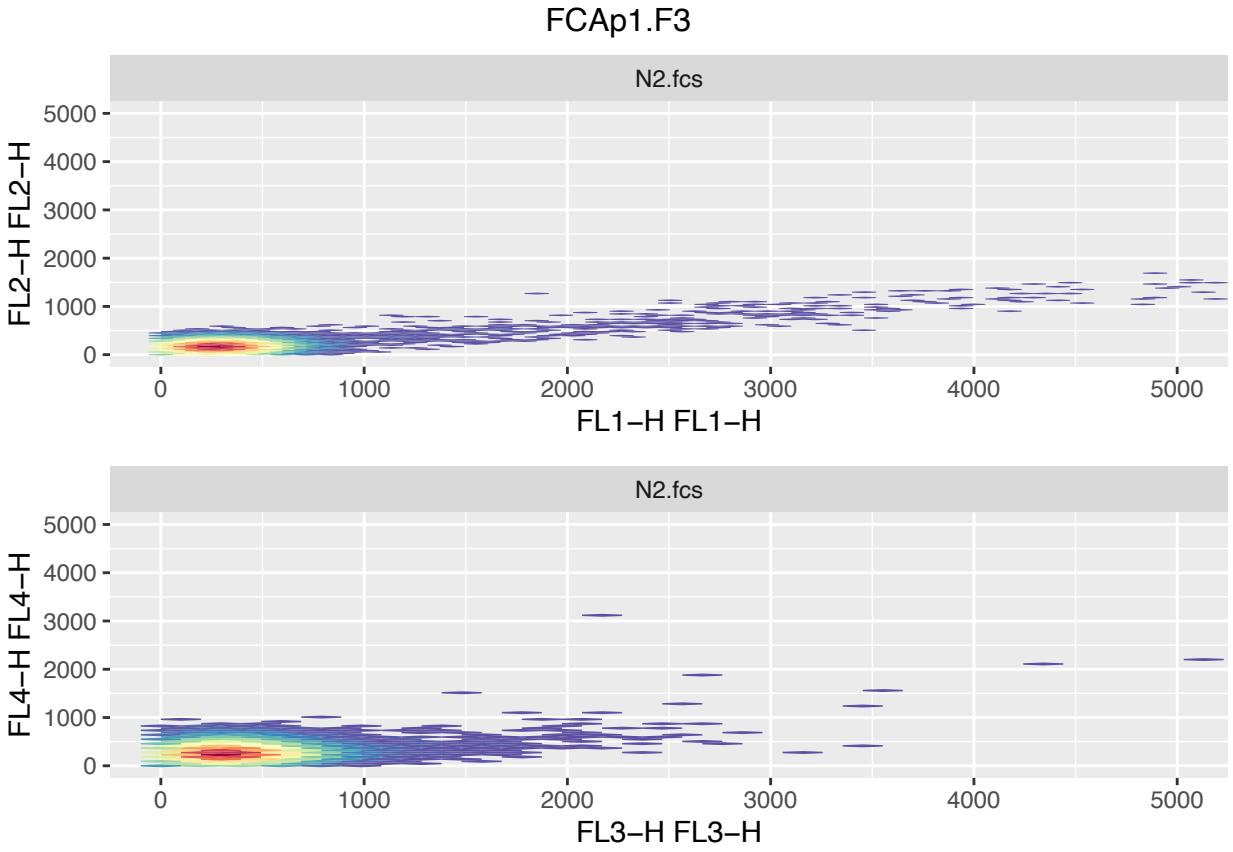

```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
ggsave(paste(dirSave,"PSB_Raw.png"),g, scale =1, dpi=300)
g
```

```
## TableGrob (2 x 2) "arrange": 3 grobs
##   z     cells    name                grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (2-2,2-2) arrange      gtable[layout]
## 3 3 (1-1,1-2) arrange text[GRID.text.374]
```

Unlabeled wild type cells

These cells are predicted to have high levels of autofluorescence

```
t <- "FCAp1.F3"
p1 <- ggcyto(ffN2.US.1,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128)  +
  coord_cartesian(xlim = c(0, 5e3), ylim = c(0, 5e3))
p2 <- ggcyto(ffN2.US.1,aes(x='FL3-H',y='FL4-H')) +
  geom_hex(bins = 128)  +
  coord_cartesian(xlim = c(0, 5e3), ylim = c(0, 5e3))
grid.arrange(as.ggplot(p1),as.ggplot(p2),top=t)
```



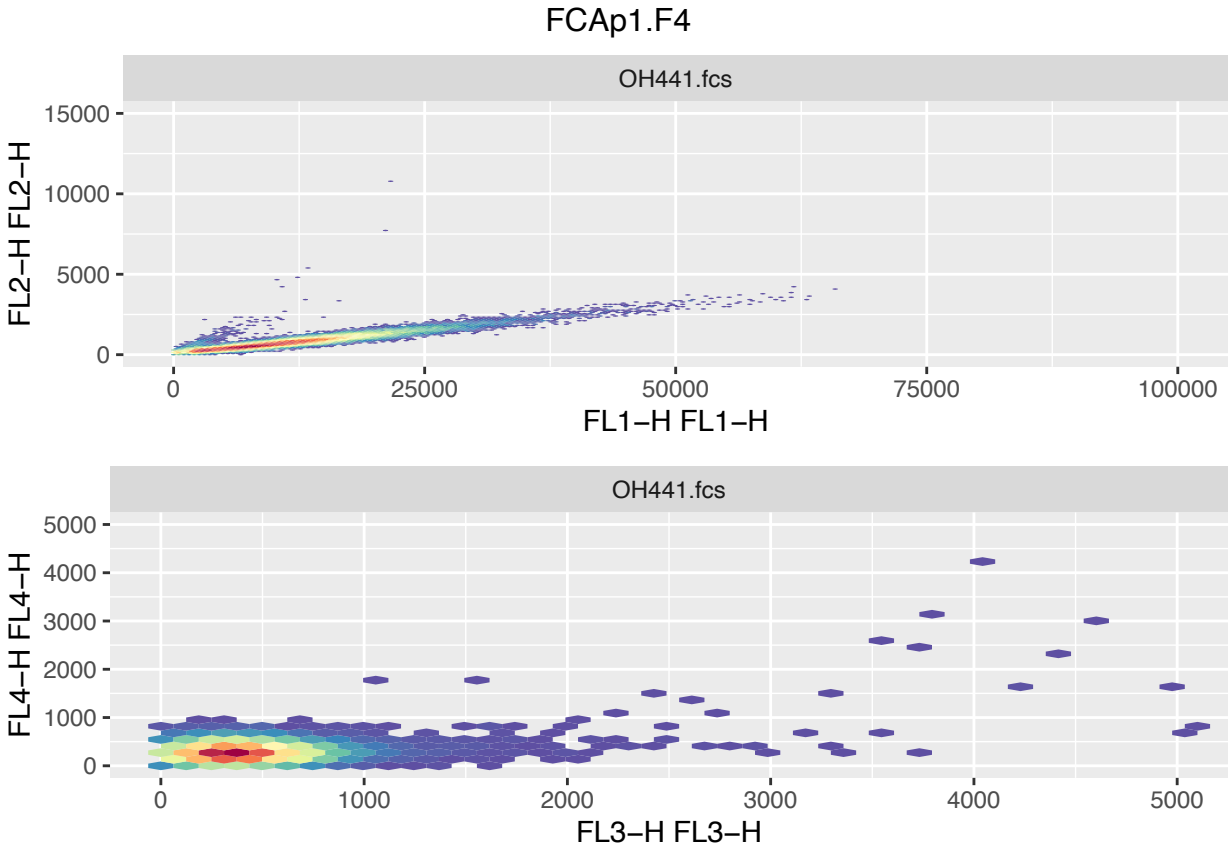

```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
ggsave(paste(dirSave,"N2_Raw.png"),g, scale =1, dpi=300)
g
```

```
## TableGrob (2 x 2) "arrange": 3 grobs
##   z     cells    name                grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (2-2,2-2) arrange      gtable[layout]
## 3 3 (1-1,1-2) arrange text[GRID.text.644]
```

unc-119p::GFP cells

GFP(+) cells are expected to be 75% of population based on {$CITE}

```
t <- "FCAp1.F4"
p1 <- ggcyto(ffGFP.US.1,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128)   +
  coord_cartesian(xlim = c(0, 1e5), ylim = c(0, 15e3))
p2 <- ggcyto(ffGFP.US.1,aes(x='FL3-H',y='FL4-H')) +
  geom_hex(bins = 128)   +
  coord_cartesian(xlim = c(0, 5e3), ylim = c(0, 5e3))
grid.arrange(as.ggplot(p1),as.ggplot(p2),top=t)
```




```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),top=t,ncol=1)
ggsave(paste(dirSave,"OH441_Raw.png"),g, scale =1, dpi=600)
g
```

```
## TableGrob (3 x 1) "arrange": 3 grobs
##   z     cells    name                grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (3-3,1-1) arrange      gtable[layout]
## 3 3 (1-1,1-1) arrange text[GRID.text.914]
```

Calibration of live/dead cutoffs from traing data sets

The LIVE/DEAD marker in use produces intense fluorescence in FL3-H when it reacts with amine groups. Dead or dying cells provide greater access to amine groups, providing a more intense signal. The algorithm must faithly discriminate weak from strong interactions

Narrow observation to 3rd quartile

In practice I have found that extremely bright events are unreliable measurements that make it challenging to evaluate underlying distributions. To obtain a more accurate understanding of these populations I am narrowing our observations space. Since I do not want to cutoff of the tail of the distribution I'm using a safety factor of 2. We can then visually confirm the impact of this filtering on the training data sets.

```
# Get the 3rd quartile value and multiple by safety factor of 2
coFL3H.NSB <- summary(ffNSB)[5,"FL3-H"]*2
coFL3H.PSB <- summary(ffPSB)[5,"FL3-H"]*2
# Generate the gating objects
gtNSB <- rectangleGate(filterId = "NSB Observation Gate",
                       "FL3-H"=c(0,coFL3H.NSB))
gtPSB <- rectangleGate(filterId = "PSB Observation Gate",
                       "FL3-H"=c(0,coFL3H.PSB))
# Gate the data to remove extreme outliers
ffNSB.1 <- Subset(ffNSB,gtNSB)
ffPSB.1 <- Subset(ffPSB,gtPSB)
```

Build data frame for dataset for quantification

Quantification is easier with a data frame is easier than flowsets

```
fl3h.NSB.1 <- exprs(ffNSB.1)[,"FL3-H"]
fl3h.PSB.1 <- exprs(ffPSB.1)[,"FL3-H"]
dfNSB.1 <- data.frame("Sample"="Non-reactive Stained Beads","FL3-H"=fl3h.NSB.1)
dfPSB.1 <- data.frame("Sample"="Reactive Stained Beads","FL3-H"=fl3h.PSB.1)
dfSB.1 <- rbind(na.omit(dfNSB.1),na.omit(dfPSB.1))
```

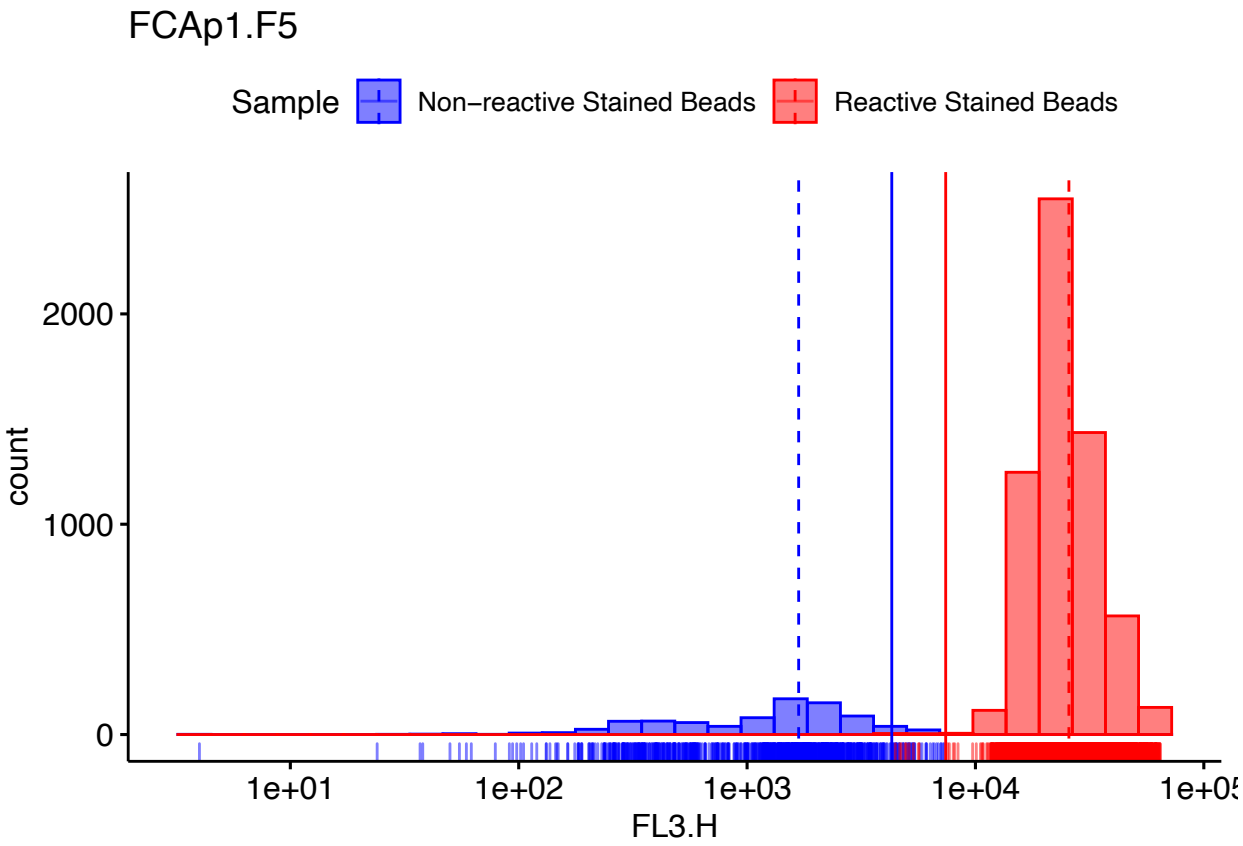
Calculate cutoff values for 'live' using standard deviation

After confirming that these populations are clearly distinguishable from one another, we will remove all non-reactive events from the reactive population to obtain a more accurate measure of population descriptives.

```
ltU <- mean(dfNSB.1$FL3.H)+2*sd(dfNSB.1$FL3.H)
gtPSB.2 <- rectangleGate(filterId = "PSB Observation Gate v2",
                       "FL3-H"=c(ltU,Inf))
ffPSB.2 <- Subset(ffPSB.1,gtPSB.2)
sdPSB.2 <- sd(exprs(ffPSB.2)[,"FL3-H"])
dtL <- mean(exprs(ffPSB.2)[,"FL3-H"])-2*sdPSB.2
```

Visualize cutoff value with adjusted reactive distribution

```
t <- "FCAp1.F5"
fl3h.PSB.2 <- exprs(ffPSB.2)[,"FL3-H"]
dfPSB.2 <- data.frame("Sample"="Reactive Stained Beads","FL3-H"=fl3h.PSB.2)
dfSB.2 <- rbind(na.omit(dfNSB.1),na.omit(dfPSB.2))
p <- gghistogram(dfSB.2, x = "FL3.H",
   add = "mean", rug = TRUE,
   color = "Sample", fill = "Sample",
   palette = c("blue", "red"))+
  scale_x_log10()+
    geom_vline(xintercept = ltU, color = "blue")+
    geom_vline(xintercept = dtL, color = "red")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.SB.FL3H.png"),p, scale =1, dpi=300)
p
```



Transfer data to a dataframe to allow easy visual inspection In this step we will also pool events from the respective .fcs files.

```
dfN2.All.S.1 <- data.frame(Sample=character(),
                    FSC.H = double(),
                    FL3.H=double(),
                    stringsAsFactors=FALSE)
dfN2.Oth.S.1 <- data.frame(Sample=character(),
                     FSC.H = double(),
                     FL3.H=double(),
                     stringsAsFactors=FALSE)
i = c(1,2)
# Combine N2 Unstained FL3H and FSCH flow frame data into one data frame
for (entry in i) {
  fl3h <- exprs(fsN2.S.1[[entry]])[,"FL3-H"]
  fsch <- exprs(fsN2.S.1[[entry]])[,"FSC-H"]
  dfT <- data.frame("Sample"="Control","FL3-H"=fl3h,"FSC-H"=fsch)
  dfN2.All.S.1 <- rbind(na.omit(dfN2.All.S.1),na.omit(dfT))
}
# Combine N2 Stained Ethanol FL3H and FSCH flow frame data into one data frame
for (entry in i) {
  fl3h <- exprs(fsN2.Eth.S.1[[entry]])[,"FL3-H"]
  fsch <- exprs(fsN2.Eth.S.1[[entry]])[,"FSC-H"]
  dfT <- data.frame("Sample"="Ethanol treated","FL3-H"=fl3h,"FSC-H"=fsch)
  dfN2.All.S.1 <- rbind(na.omit(dfN2.All.S.1),na.omit(dfT))
}
fl3h <- exprs(fsN2.Oth.S.1[[1]])[,"FL3-H"]
fsch <- exprs(fsN2.Oth.S.1[[1]])[,"FSC-H"]
```

```
dfT <- data.frame("Sample"="Puromycin 100ug/ml","FL3-H"=fl3h,"FSC-H"=fsch)
dfN2.All.S.1 <- rbind(dfN2.All.S.1,na.omit(dfT))
fl3h <- exprs(fsN2.Oth.S.1[[2]])[,"FL3-H"]
fsch <- exprs(fsN2.Oth.S.1[[2]])[,"FSC-H"]
dfT <- data.frame("Sample"="Heat treated","FL3-H"=fl3h,"FSC-H"=fsch)
dfN2.All.S.1 <- rbind(dfN2.All.S.1,na.omit(dfT))

# This is a work around for making a second plot of just these samples
fl3h <- exprs(fsN2.Oth.S.1[[1]])[,"FL3-H"]
fsch <- exprs(fsN2.Oth.S.1[[1]])[,"FSC-H"]
dfN2.PR.S.1 <- data.frame("Sample"="Puromycin 100ug/ml","FSC-H"=fsch, "FL3-H"=fl3h)
dfN2.Oth.S.1 <- rbind(dfN2.Oth.S.1,na.omit(dfN2.PR.S.1))
fl3h <- exprs(fsN2.Oth.S.1[[2]])[,"FL3-H"]
fsch <- exprs(fsN2.Oth.S.1[[2]])[,"FSC-H"]
dfN2.HKC.S.1 <- data.frame("Sample"="Heat treated","FSC-H"=fsch, "FL3-H"=fl3h)
dfN2.Oth.S.1 <- rbind(dfN2.Oth.S.1,na.omit(dfN2.HKC.S.1))
```

Estimating lower bound for dead cutoff from lethal treated cells

```
qD.N2.PR.S.1 <- quantile(dfN2.PR.S.1$FL3.H, probs = .3)
print("FL3H percentile cutoff for stained puro treated N2")
```

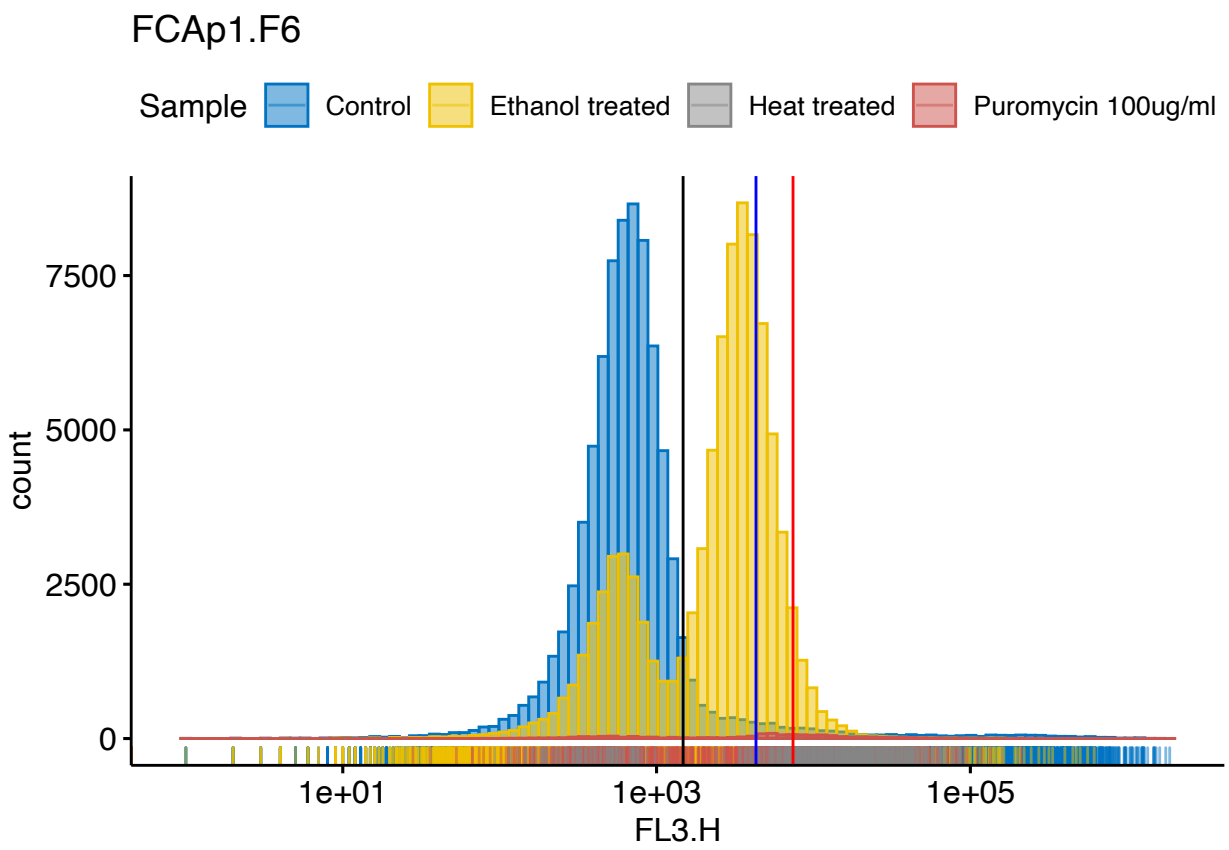
```
## [1] "FL3H percentile cutoff for stained puro treated N2"
```

```
qD.N2.PR.S.1
```

```
##    30%
## 1476.4
```

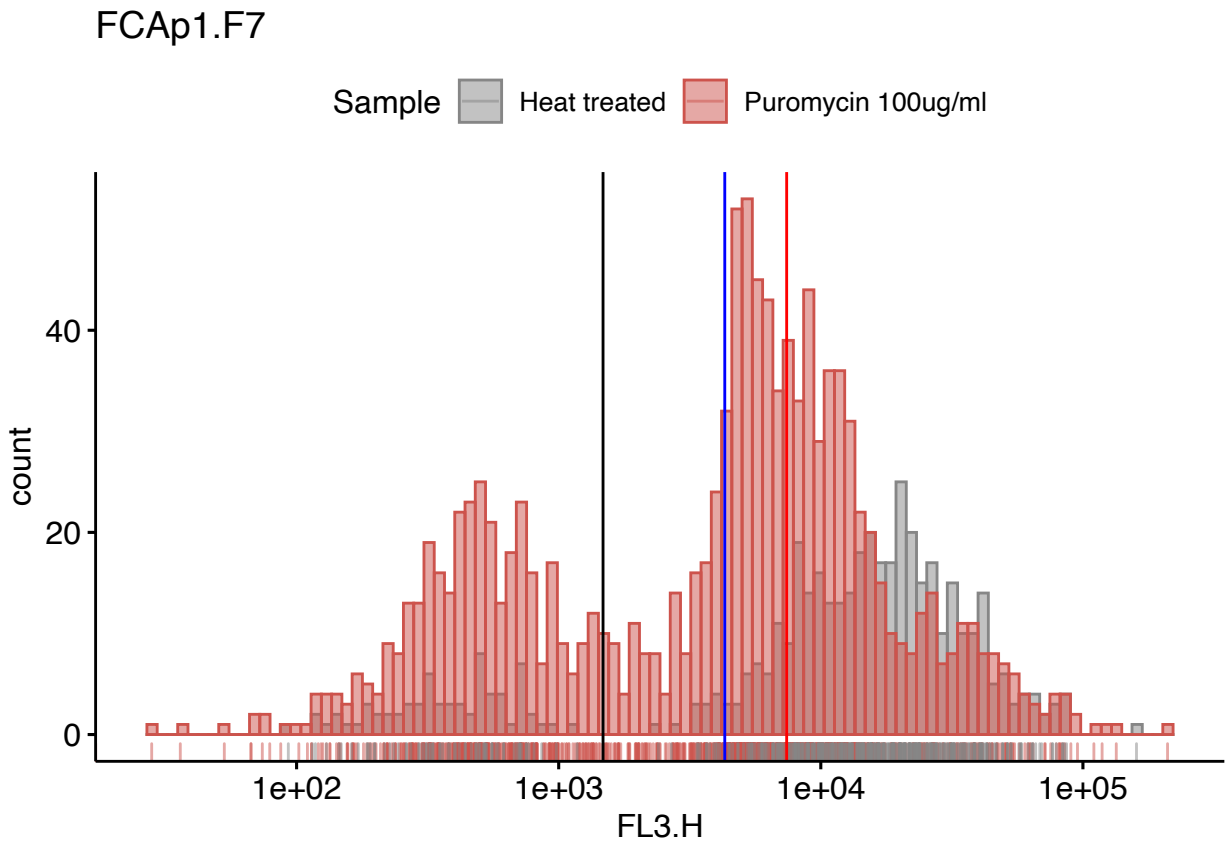
Visualize all stained N2 data

```
t <- "FCAp1.F6"
p <- gghistogram(dfN2.All.S.1, x = "FL3.H", rug = TRUE,
                  color = "Sample", fill = "Sample",
                  palette = "jco", bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = dtL, color = "red")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfN2.All.S.1.FL3H.png"),p, scale =1, dpi=300)
p
```



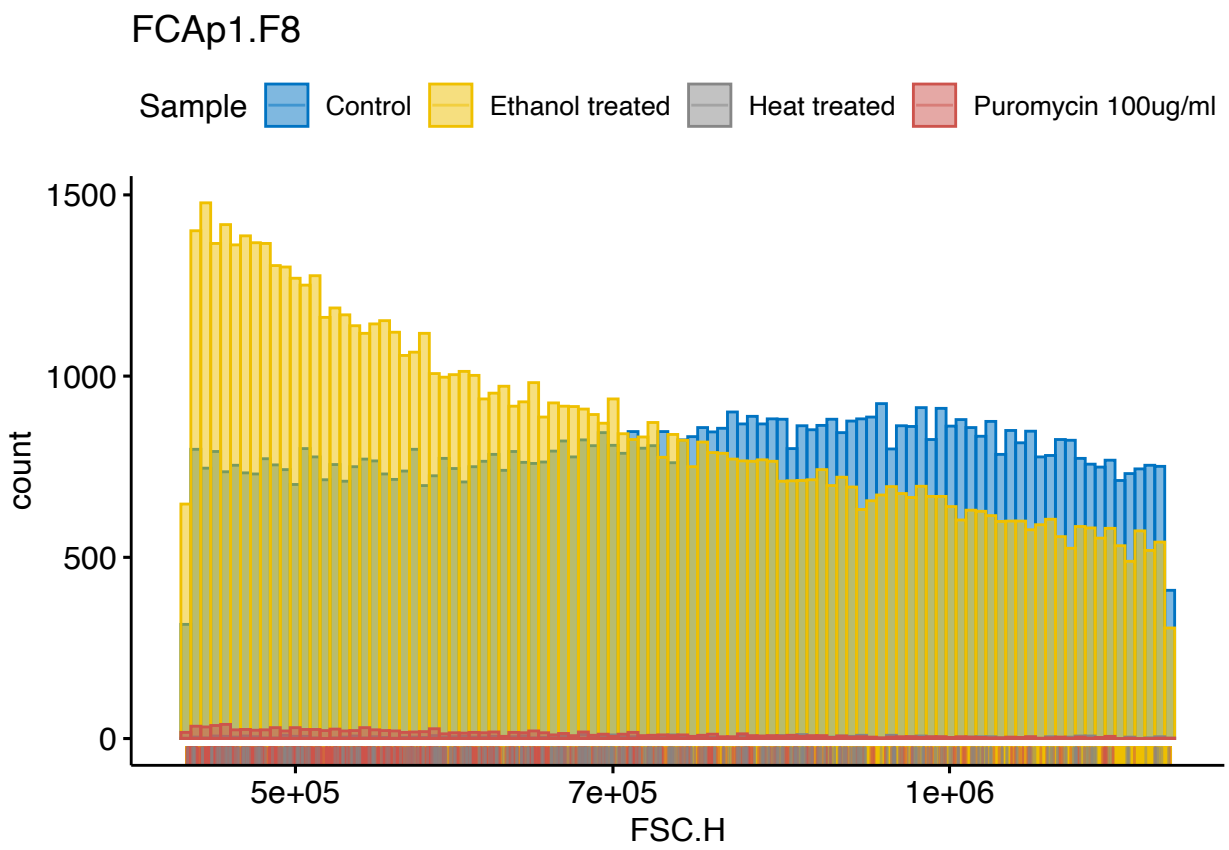
Visualize only Puro and HKC stained N2 data

```
t <- "FCAp1.F7"
p <- gghistogram(dfN2.Oth.S.1, x = "FL3.H", rug = TRUE,
                        color = "Sample", fill = "Sample",
                        palette = c("#868686FF","#CD534CFF"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = dtL, color = "red")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfN2.Oth.S.1.FL3H.png"),p, scale =1, dpi=300)
p
```



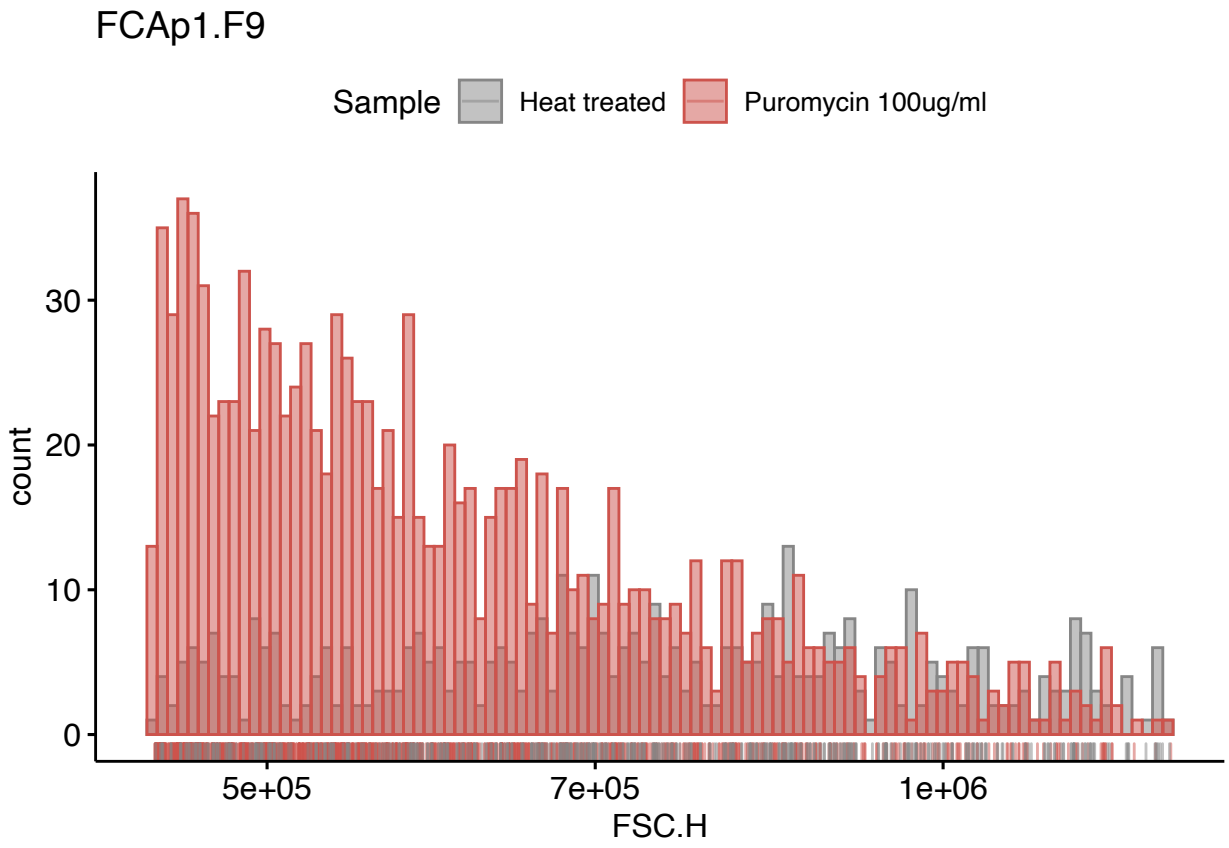
Check FSC-H values fo all stained N2 samples

```
t <- "FCAp1.F8"
p <- gghistogram(dfN2.All.S.1, x = "FSC.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = "jco", bins = 100)+
  scale_x_log10()+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfN2.All.S.1.FSCH.png"),p, scale =1, dpi=300)
p
```



Check FSC-H values for only Puro and HKC stained N2 samples

```
t <- "FCAp1.F9"
p <- gghistogram(dfN2.Oth.S.1, x = "FSC.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("#868686FF","#CD534CFF"), bins = 100)+
  scale_x_log10()+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.N2.dfN2.Oth.S.1.FSCH.png"),p, scale =1, dpi=300)
p
```



Optimization of GFP detection via autofluorescence modeling

Generate the linear model for autofluoresence

```
# First generate the linear model from the N2 data set. To do this the event
# values must be extracted from the flow frame and be arranged into a dataframe
fl1h.N2.US.1 <- exprs(ffN2.US.1)[,"FL1-H"]
fl2h.N2.US.1 <- exprs(ffN2.US.1)[,"FL2-H"]
fl3h.N2.US.1 <- exprs(ffN2.US.1)[,"FL3-H"]
dfN2.US.1 <- data.frame("FL1-H" = fl1h.N2.US.1,"FL2-H"=fl2h.N2.US.1,"FL3-H"=fl3h.N2.US.1)
lmAF.12 <- lm(FL2.H~FL1.H, data = dfN2.US.1)
lmAF.12.int = lmAF.12$coefficients[[1]]      # Vertical intercept
lmAF.12.m = lmAF.12$coefficients[[2]]        # Slope
print("Summary information for autofluoresence linear model")
```

```
## [1] "Summary information for autofluoresence linear model"
```

```
summary(lmAF.12)
```

```
##
## Call:
## lm(formula = FL2.H ~ FL1.H, data = dfN2.US.1)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -4840.3   -66.4    -3.0    62.3  5289.5
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.080e+02  2.283e-01   473.1   <2e-16 ***
## FL1.H       2.251e-01  4.544e-04   495.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.17 on 296983 degrees of freedom
## Multiple R-squared:  0.4525, Adjusted R-squared:  0.4525
## F-statistic: 2.455e+05 on 1 and 296983 DF,  p-value: < 2.2e-16
```

Generate a polygon gate to select GFP(+) subpopulation

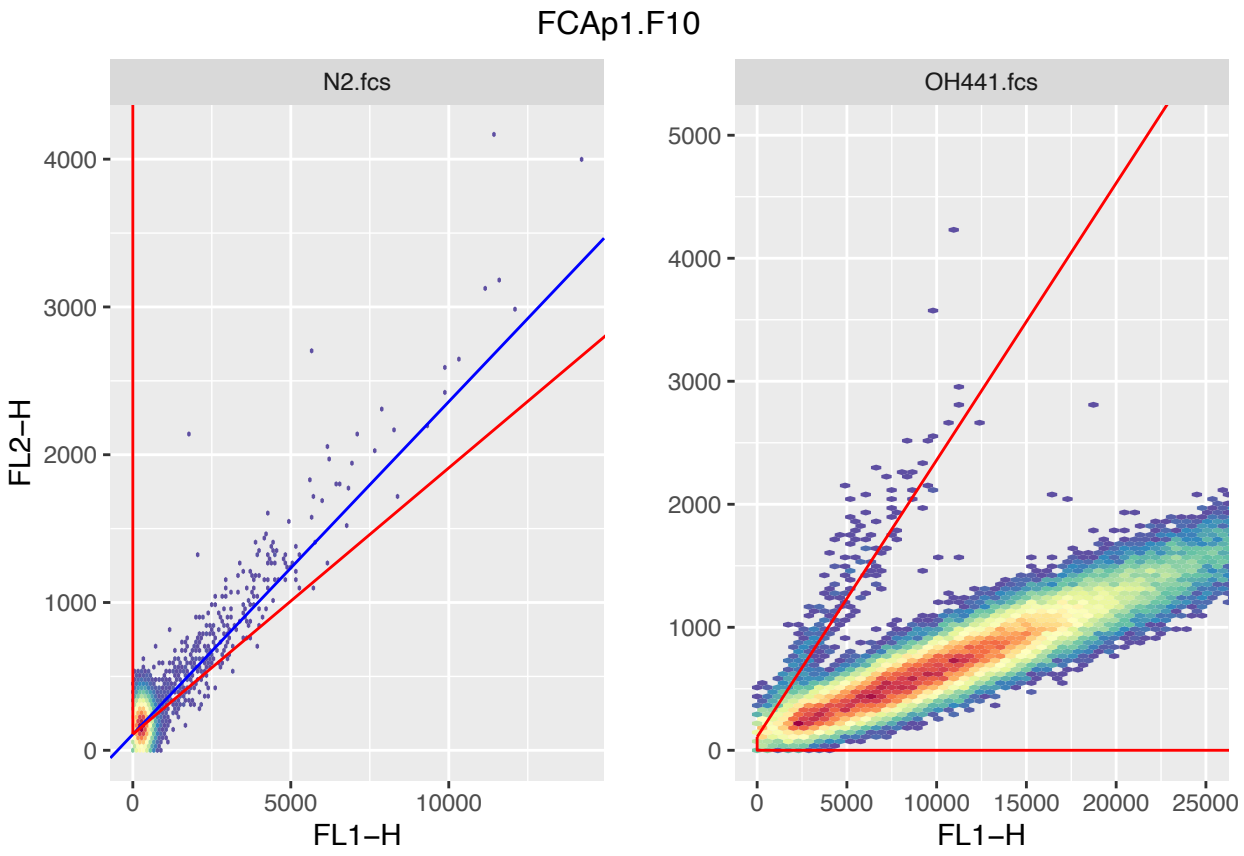
Since the GFP(+) data set as a known subpopulation of GFP(-) cells, I will use linear regression based on N2 cells to predict which events need to be removed from the GFP(+) training data set.

```
# Calculate coordinates for polygon gate using lmAF
stdErr <- summary(lmAF.12)$coefficients[1,2]
intU <- lmAF.12.int+2*stdErr
intL <- lmAF.12.int-2*stdErr
m <-lmAF.12.m
maxFL1H = 1.01*summary(ffGFP.US.1)[6,'FL1-H']
maxFL2H = 1.01*summary(ffGFP.US.1)[6,'FL2-H']
xU = (maxFL2H-intL)/m

# Define the non-autofluorescence polygon gate based on coordinates
v1 = c(0,intU)
v2 = c(xU,maxFL2H)
v3 = c(maxFL1H,maxFL2H)
v4 = c(maxFL1H,0)
v5 = c(0,0)
verts <- rbind(v1,v2,v3,v4,v5)
colnames(verts) <- c("FL1-H","FL2-H")
gtNAF <- polygonGate(filterId="Non-Autofluorescence", .gate = verts)

# Repeat the process to make the inverse gate
v1 = c(0,intU)
v2 = c(0,maxFL2H)
v3 = c(maxFL1H,maxFL2H)
verts <- rbind(v1,v2,v3)
colnames(verts) <- c("FL1-H","FL2-H")
gtAF <- polygonGate(filterId="Autofluorescence", .gate = verts)
```

Visualize linear model and population pre gating

```
t <- "FCAp1.F10"
set.seed(5)
p1 <- ggcyto(ffN2.US.1,aes(x='FL1-H',y='FL2-H')) +
```

```
  geom_hex(bins = 128) +
  geom_abline(slope = lmAF.12.m, intercept = lmAF.12.int, color = "blue")+
  labs(x ="FL1-H", y = "FL2-H")+
  geom_gate(gtAF)
p2 <- ggcyto(ffGFP.US.1,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  coord_cartesian(xlim = c(0, 25e3), ylim = c(0, 5e3))+
  labs(x ="FL1-H", y = "")+
  geom_gate(gtNAF)
grid.arrange(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
```




```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
ggsave(paste(dirSave,"2DHist.N2vGFP.US.1.PreAFGating.WLines.png"),g, scale =1, dpi=300)
```

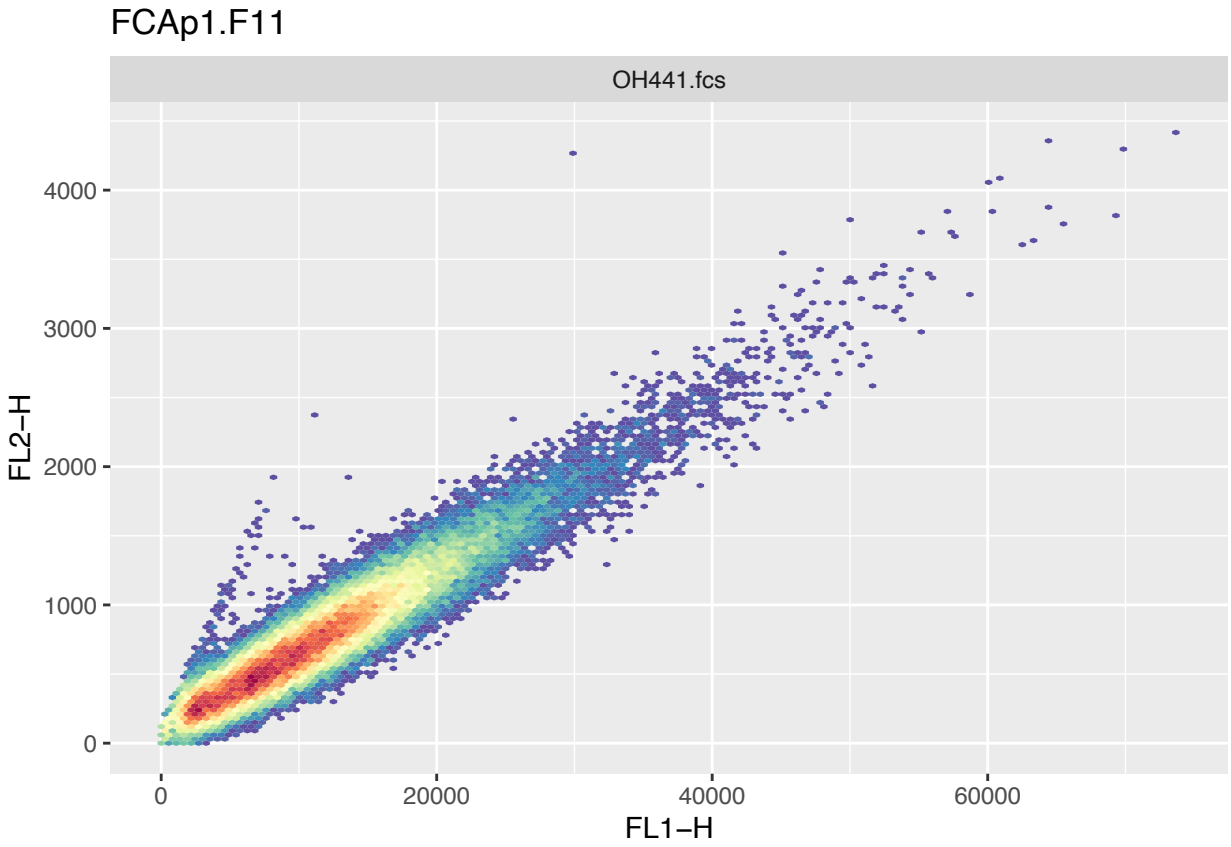
Gate unc-119::GFP cells for autofluorescent events

```
ffGFP.US.2 <- Subset(ffGFP.US.1, gtNAF)
```

Visualize changes resulting from AF gating

```
set.seed(5)
t <- "FCAp1.F11"
```

```
p <- ggcyto(ffGFP.US.2,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  labs(x ="FL1-H", y = "FL2-H")+
  ggtitle(t)
ggsave(paste(dirSave,"2DHist.GFP.US.2.PostAFGating.NoLines.png"),p, scale =1, dpi=300)
p
```



Transfer data to data frame for plotting

```
fl1h.GFP.US.2 <- exprs(ffGFP.US.2)[,"FL1-H"]
fl1h.N2.US.1 <- exprs(ffN2.US.1)[,"FL1-H"]
dfGFP.US.2 <- data.frame("Sample"="OH441", "FL1-H" = fl1h.GFP.US.2)
dfN2.US.1 <- data.frame("Sample"="N2", "FL1-H" = fl1h.N2.US.1)
dfGFPvN2.US <- rbind(dfGFP.US.2,dfN2.US.1)
```

Estimating lower bound for GFP cutoff

```
qGFP.US.2 <- quantile(dfGFP.US.2$FL1.H, probs = .015)
print("FL1H percentile cutoff for unstained OH441")
```

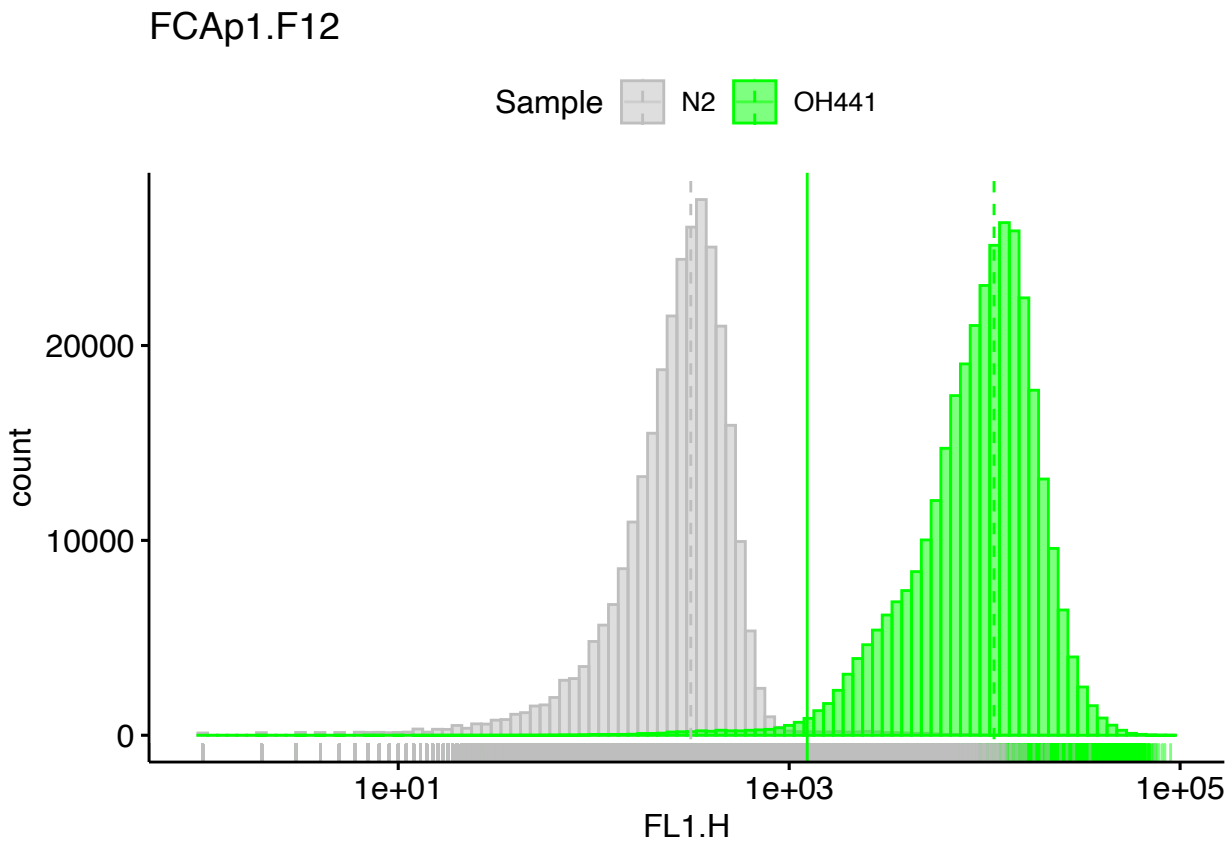
```
## [1] "FL1H percentile cutoff for unstained OH441"
```

```
qGFP.US.2
```

```
## 1.5%
## 1237
```

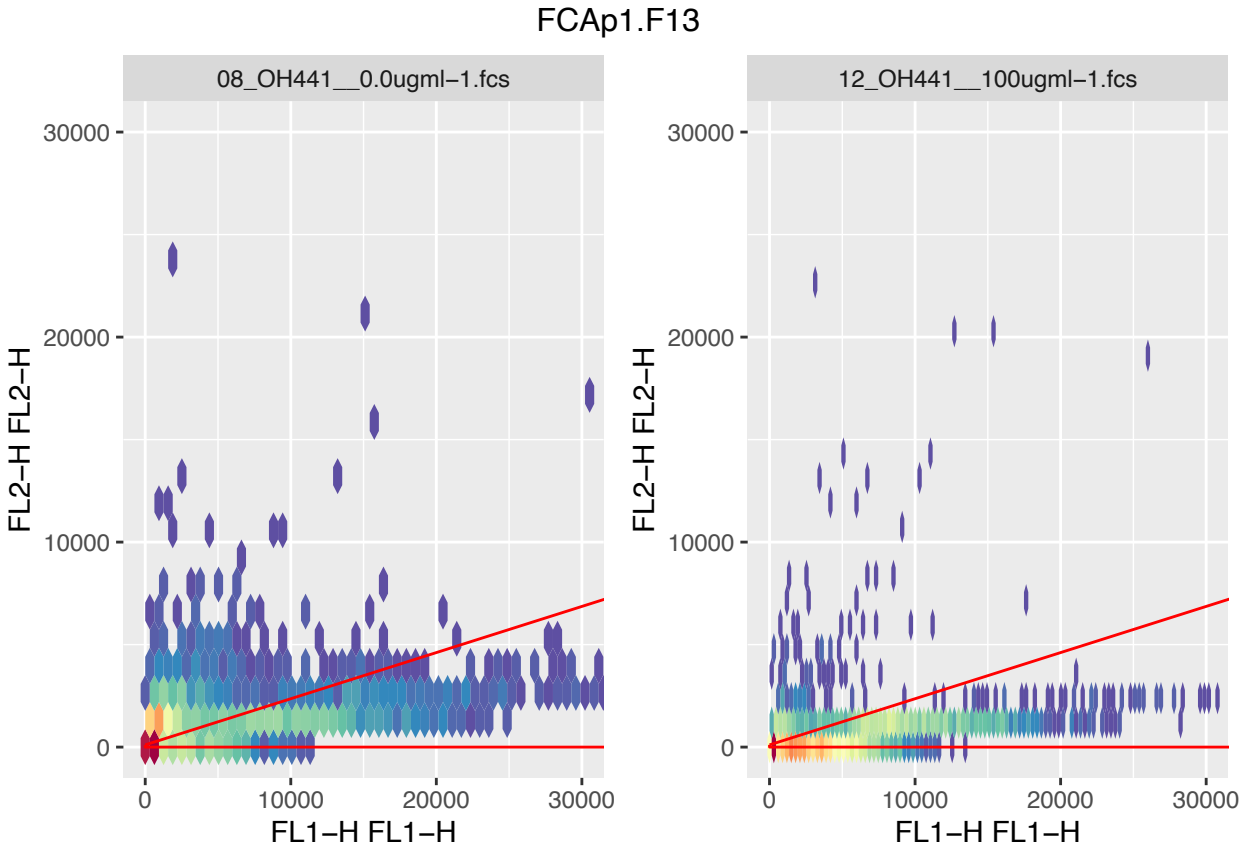
Visualize unstained N2 and GFP(+) distributions following gating

```
t <- "FCAp1.F12"
p <- gghistogram(dfGFPvN2.US, x = "FL1.H", rug = TRUE,
                      add = "mean",color = "Sample", fill = "Sample",
                      palette = c("Grey","green"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfGFPvN2.US.png"),p, scale =1, dpi=300)
p
```



Does amine reactive dye alter the appearance of GFP(+) cells? ### Visualize linear model and population pre gating

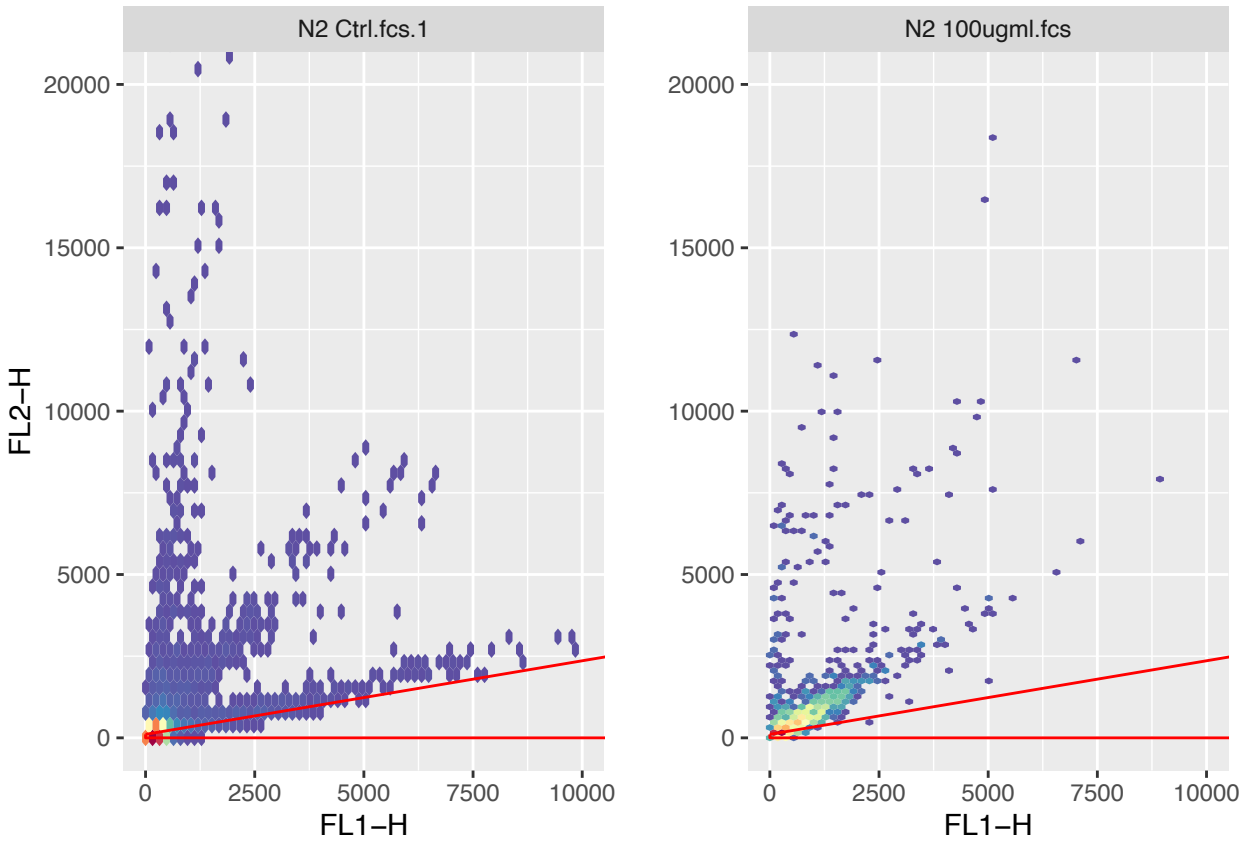
```
set.seed(5)
t <- "FCAp1.F13"
p1 <- ggcyto(ffGFP.SL.1,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 256) +
  coord_cartesian(xlim = c(0, 30e3), ylim = c(0, 30e3))+
  geom_gate(gtNAF)
p2 <- ggcyto(ffGFP.SP.1,aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  coord_cartesian(xlim = c(0, 30e3), ylim = c(0, 30e3))+
  geom_gate(gtNAF)
grid.arrange(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
```



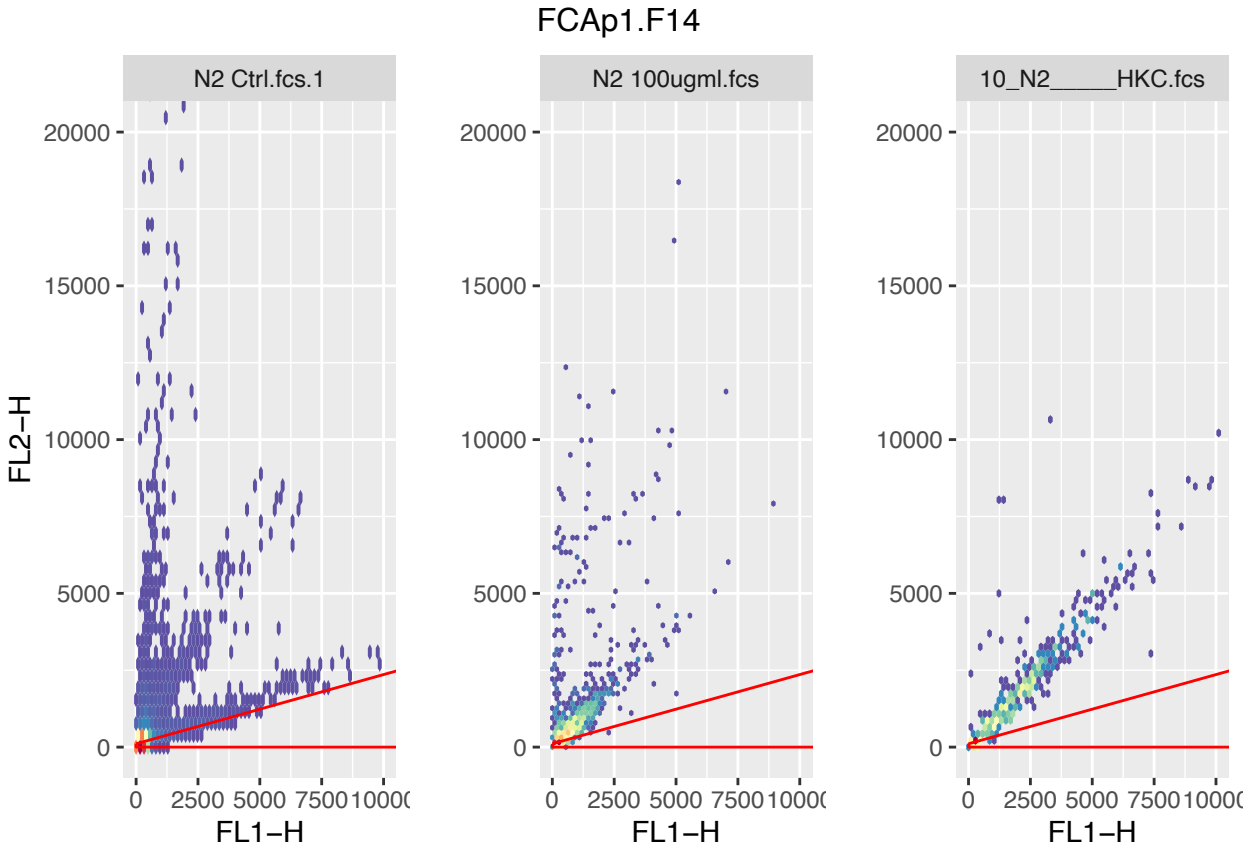

```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),top=t,ncol=2)
ggsave(paste(dirSave,"2DHist.ffGFP.SLvSP.1.png"),g, scale =1, dpi=300)
```

Does the AF gate accurately exclude stained N2 cells?

```
set.seed(5)
t <- "FCAp1.F14"
p1 <- ggcyto(fsN2.S.1[[2]],aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  coord_cartesian(xlim = c(0, 10e3), ylim = c(0, 20e3))+
  labs(x ="FL1-H", y = "FL2-H") +
  geom_gate(gtNAF)
p2 <- ggcyto(fsN2.Oth.S.1[[1]],aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  labs(x ="FL1-H", y = "")+
  coord_cartesian(xlim = c(0, 10e3), ylim = c(0, 20e3))+
  geom_gate(gtNAF)
grid.arrange(as.ggplot(p1),as.ggplot(p2),ncol=2)
```




```
p3 <- ggcyto(fsN2.Oth.S.1[[2]],aes(x='FL1-H',y='FL2-H')) +
  geom_hex(bins = 128) +
  labs(x ="FL1-H", y = "")+
  coord_cartesian(xlim = c(0, 10e3), ylim = c(0, 20e3))+
  geom_gate(gtNAF)
grid.arrange(as.ggplot(p1),as.ggplot(p2),as.ggplot(p3),top=t,ncol=3)
```




```
g <- arrangeGrob(as.ggplot(p1),as.ggplot(p2),as.ggplot(p3),top=t,ncol=3)
ggsave(paste(dirSave,"2DHist.ffN2.CtrlvOth.1.png"),g, scale =3, dpi=300)
```

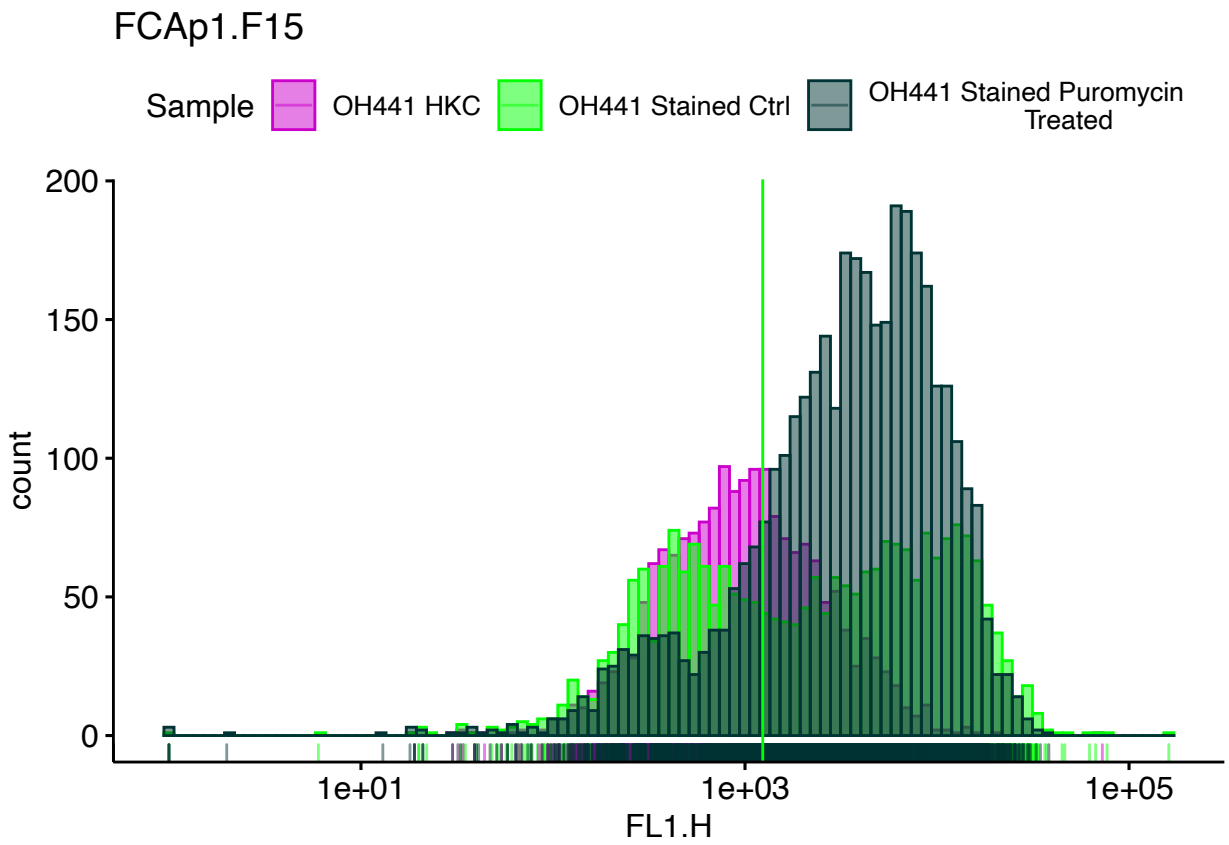
Transfer data to a dataframe to allow easy visual inspection

GFP.SL = OH441 Stained with live/dead, no lethal treatment GFP.SD = OH441 Stained with live/dead, heat treatment GFP.SP = OH441 Stained with live/dead, high puro treatment

```
fl1h.SL.1 <- exprs(ffGFP.SL.1)[,"FL1-H"]
fl3h.SL.1 <- exprs(ffGFP.SL.1)[,"FL3-H"]
dfSL.1 <- data.frame("Sample"="OH441 Stained Ctrl","FL1-H"=fl1h.SL.1,
                     "FL3-H"=fl3h.SL.1)

fl1h.SD.1 <- exprs(ffGFP.SD.1)[,"FL1-H"]
fl3h.SD.1 <- exprs(ffGFP.SD.1)[,"FL3-H"]
dfSD.1 <- data.frame("Sample"="OH441 HKC","FL1-H"=fl1h.SD.1,
                     "FL3-H"=fl3h.SD.1)

fl1h.SP.1 <- exprs(ffGFP.SP.1)[,"FL1-H"]
fl3h.SP.1 <- exprs(ffGFP.SP.1)[,"FL3-H"]
dfSP.1 <- data.frame("Sample"="OH441 Stained Puromycin
                     Treated","FL1-H"=fl1h.SP.1,"FL3-H"=fl3h.SP.1)

dfGFP.SLDP.1 <- rbind(dfSL.1,dfSD.1,dfSP.1)
```

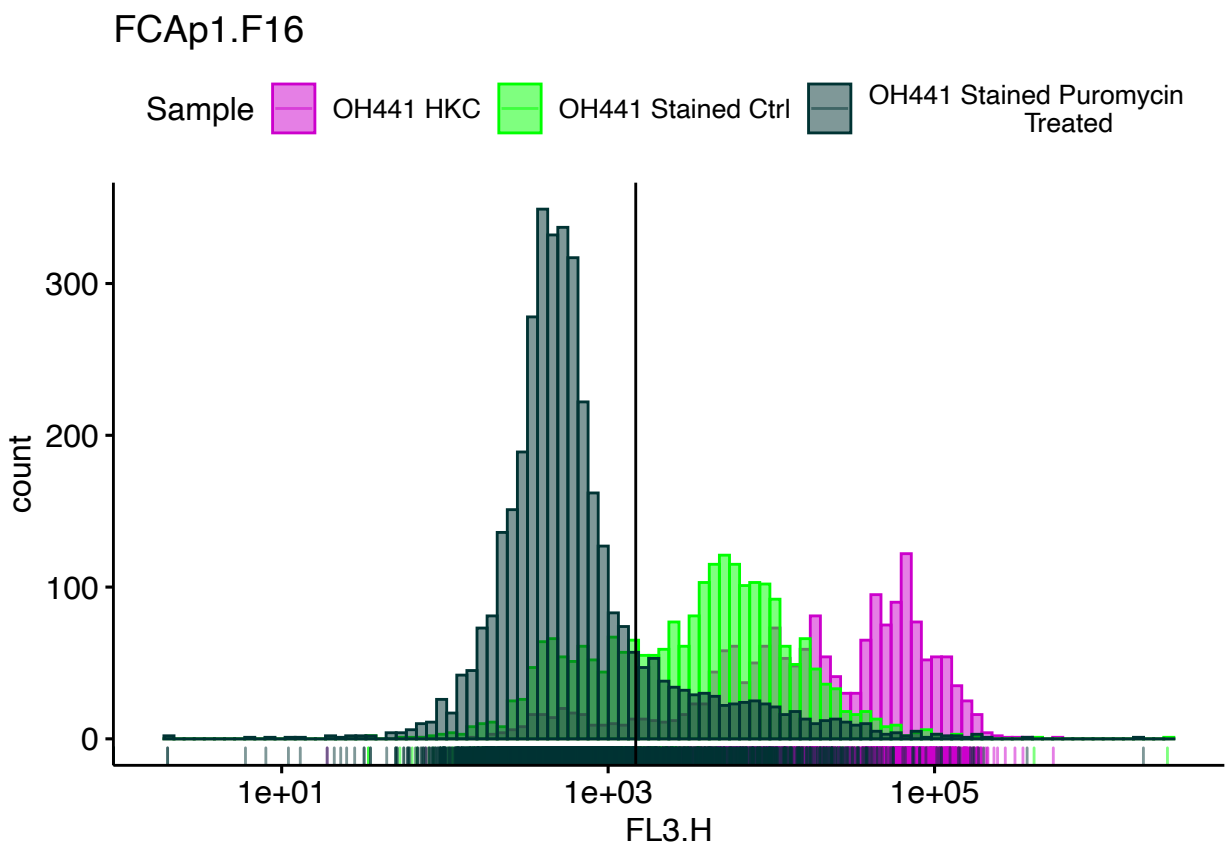
Visualize distributions of GFP(+) cells pre AF gating in FL1H

```
t <- "FCAp1.F15"
p <- gghistogram(dfGFP.SLDP.1 , x = "FL1.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("#CC00CC","green","#003333"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfGFP.SLDP.1.FL1.png"),p, scale =1, dpi=300)
p
```



Visualize distributions of GFP(+) cells pre AF gating in FL3H These cells were stained with propidium iodide

```
t <- "FCAp1.F16"
p <- gghistogram(dfGFP.SLDP.1, x = "FL3.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("#CC00CC","green","#003333"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfGFP.SLDP.1.FL3.png"),p, scale =1, dpi=300)
p
```



Gate stained unc-119::GFP cells for autofluorescent events

```
ffGFP.SL.2 <- Subset(ffGFP.SL.1, gtNAF)
ffGFP.SD.2 <- Subset(ffGFP.SD.1, gtNAF)
ffGFP.SP.2 <- Subset(ffGFP.SP.1, gtNAF)
```

Transfer data to a dataframe to allow easy visual inspection

In this step we will also pool events from the respective .fcs files.

```
fl1h.SL.2 <- exprs(ffGFP.SL.2)[,"FL1-H"]
fl3h.SL.2 <- exprs(ffGFP.SL.2)[,"FL3-H"]
dfSL.2 <- data.frame("Sample"="OH441 Stained Ctrl","FL1-H"=fl1h.SL.2,"FL3-H"=fl3h.SL.2)

fl1h.SD.2 <- exprs(ffGFP.SD.2)[,"FL1-H"]
fl3h.SD.2 <- exprs(ffGFP.SD.2)[,"FL3-H"]
dfSD.2 <- data.frame("Sample"="OH441 HKC","FL1-H"=fl1h.SD.2,"FL3-H"=fl3h.SD.2)

fl1h.SP.2 <- exprs(ffGFP.SP.2)[,"FL1-H"]
fl3h.SP.2 <- exprs(ffGFP.SP.2)[,"FL3-H"]
dfSP.2 <- data.frame("Sample"="OH441 Stained Puromycin
                     Treated","FL1-H"=fl1h.SP.2,"FL3-H"=fl3h.SP.2)

dfGFP.SLDP.2 <- rbind(dfSL.2,dfSD.2,dfSP.2)
```

Estimating lower bound for dead cutoff from lethal treated cells

```
qGFP.SP.2 <- quantile(dfSP.2$FL3.H, probs = .6)
print("FL3H percentile cutoff for stained puro treated OH441")
```

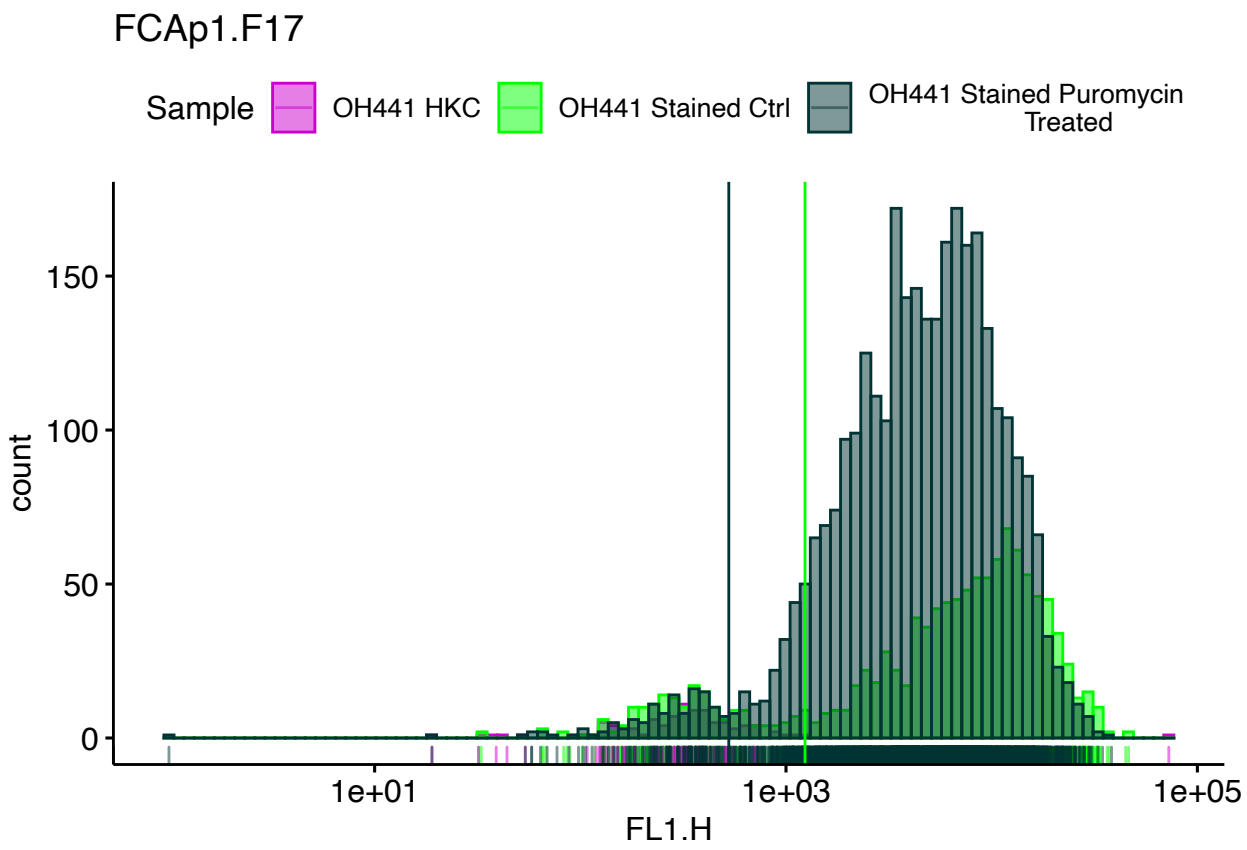
```
## [1] "FL3H percentile cutoff for stained puro treated OH441"
```

```
qGFP.SP.2
```

```
## 60%
## 527
```

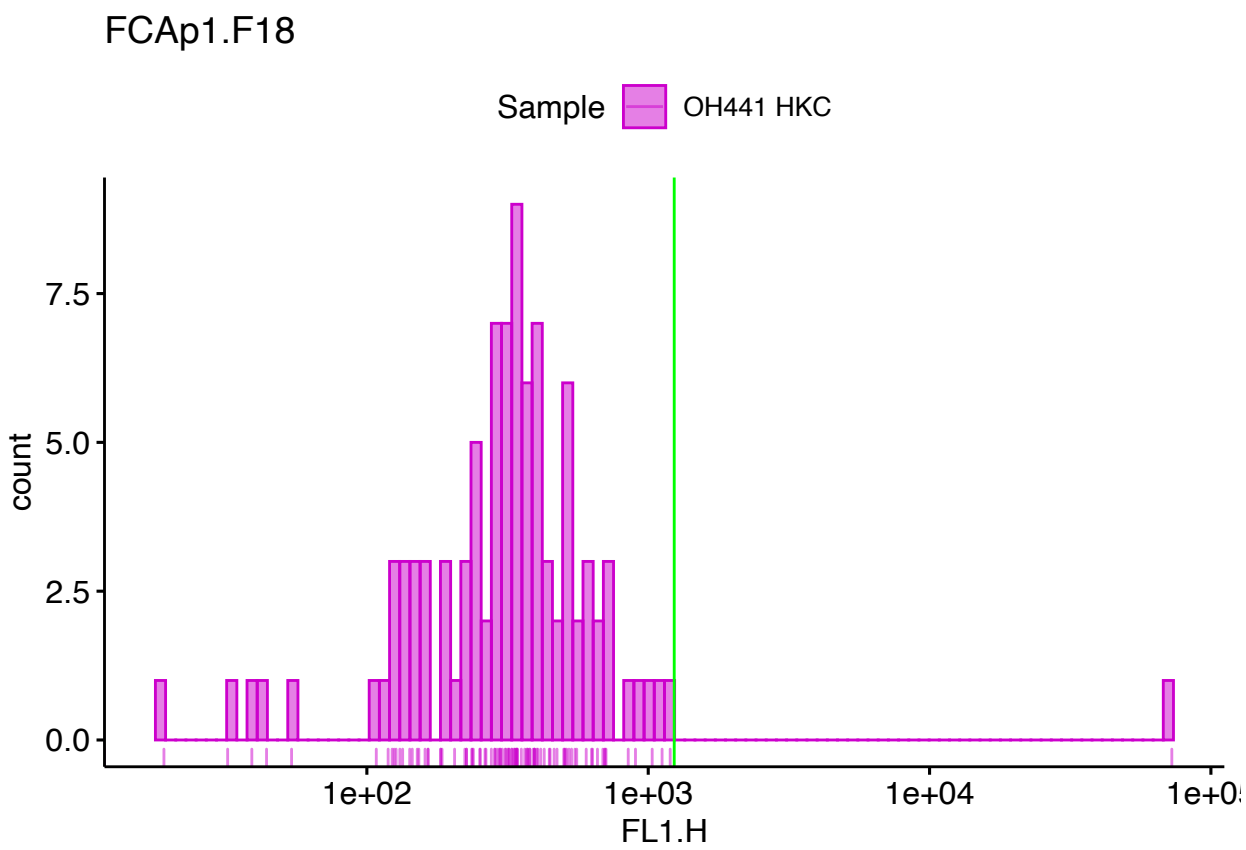
Visualize distributions of GFP(+) cells post AF gating

```
t <- "FCAp1.F17"
p <- gghistogram(dfGFP.SLDP.2 , x = "FL1.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("#CC00CC","green","#003333"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  geom_vline(xintercept = qGFP.SP.2, color = "#003333")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfGFP.SLDP.2.FL1H.png"),p, scale =1, dpi=300)
p
```



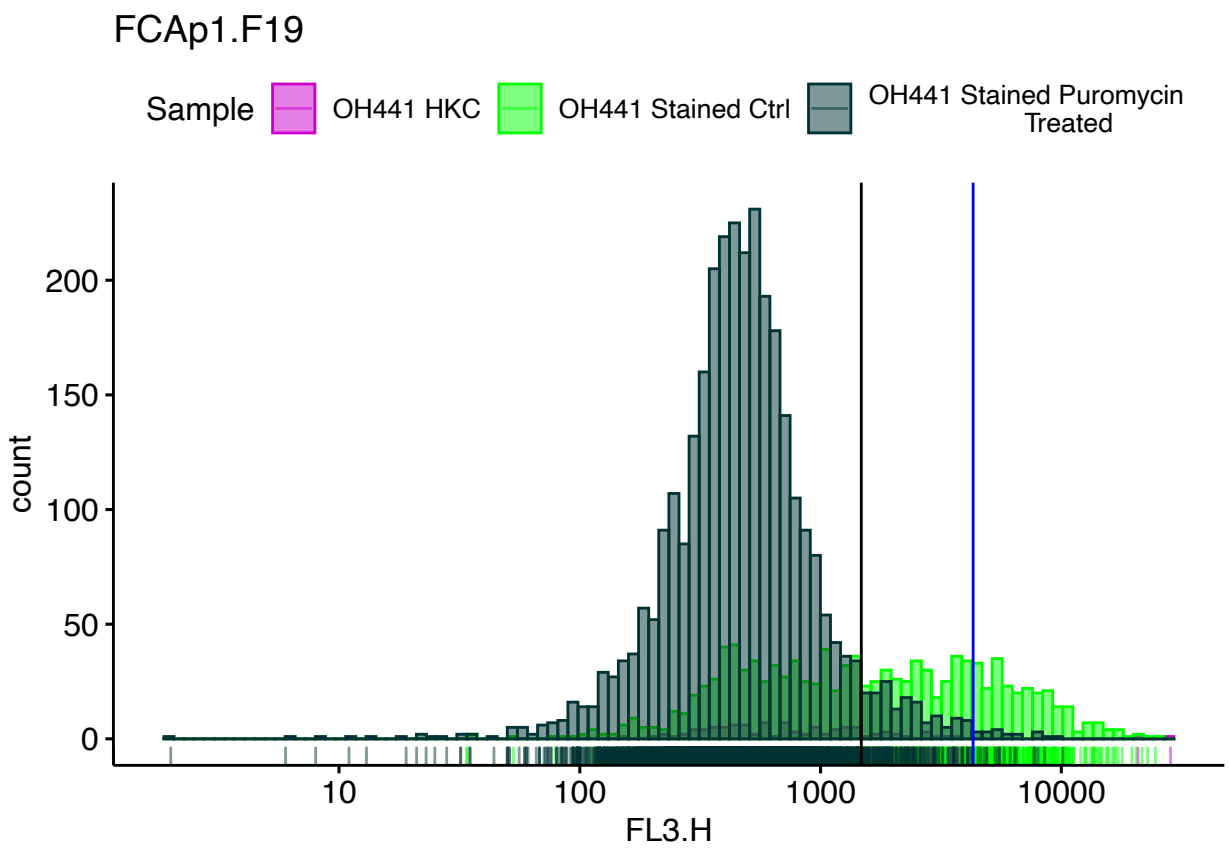
Visualize just HKC

```
t <- "FCAp1.F18"
p <- gghistogram(dfSD.2 , x = "FL1.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = "#CC00CC", bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfSD.2.FL1H.png"),p, scale =1, dpi=300)
p
```



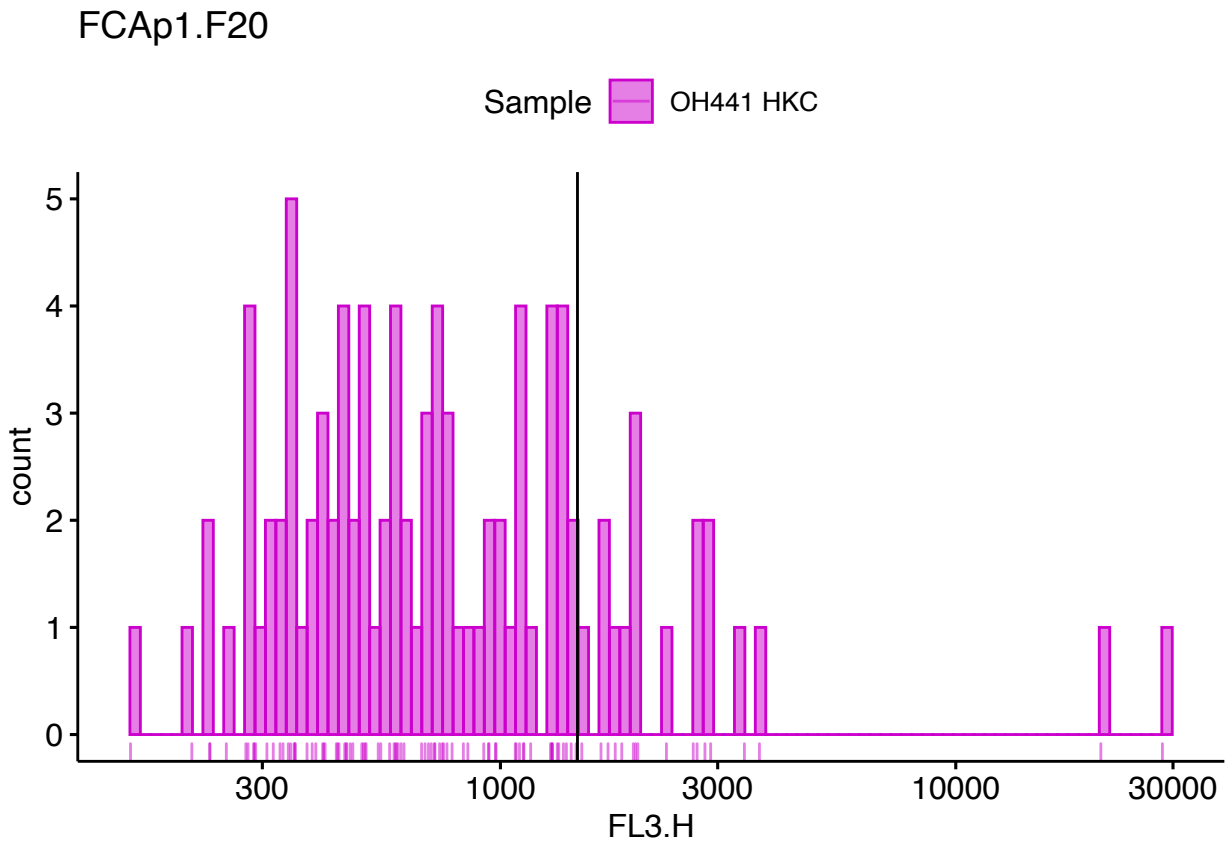
Visualize distributions of GFP(+) cells post AF gating in FL3H

```
t <- "FCAp1.F19"
p <- gghistogram(dfGFP.SLDP.2 , x = "FL3.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("#CC00CC","green","#003333"), bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  geom_vline(xintercept = ltU, color = "blue")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfGFP.SLDPR.2.FL3H.png"),p, scale =1, dpi=300)
p
```



Visualize just HKC

```
t <- "FCAp1.F20"
p <- gghistogram(dfSD.2 , x = "FL3.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = "#CC00CC", bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfSD.2.FL3H.png"),p, scale =1, dpi=300)
p
```



What happens if you only compensate the data but do not AF gate?

Build compensation matrix

```
fsComp <- flowSet(
  ffN2.US.1[,c("FSC-H","SSC-H","FL1-H","FL2-H","FL3-H")],
  ffGFP.US.2[,c("FSC-H","SSC-H","FL1-H","FL2-H","FL3-H")],
  ffOFP.US.1[,c("FSC-H","SSC-H","FL1-H","FL2-H","FL3-H")],
  ffPSB.1[,c("FSC-H","SSC-H","FL1-H","FL2-H","FL3-H")])
spillmat <- spillover(fsComp, unstained=1, fsc="FSC-H", ssc="SSC-H",
                      stain_match="ordered")
```

Compensate the data

For GFP samples use .1 since .2 files are post AF gating

```
fsN2.S.2 <- compensate(fsN2.S.1,spillmat)
ffGFP.SL.3 <- compensate(ffGFP.SL.1,spillmat)
ffGFP.SD.3 <- compensate(ffGFP.SD.1,spillmat)
ffGFP.SP.3 <- compensate(ffGFP.SL.1,spillmat)
```

Write spillover matrix to file

```
write.csv(spillmat, file = fnComp, row.names = TRUE)
```

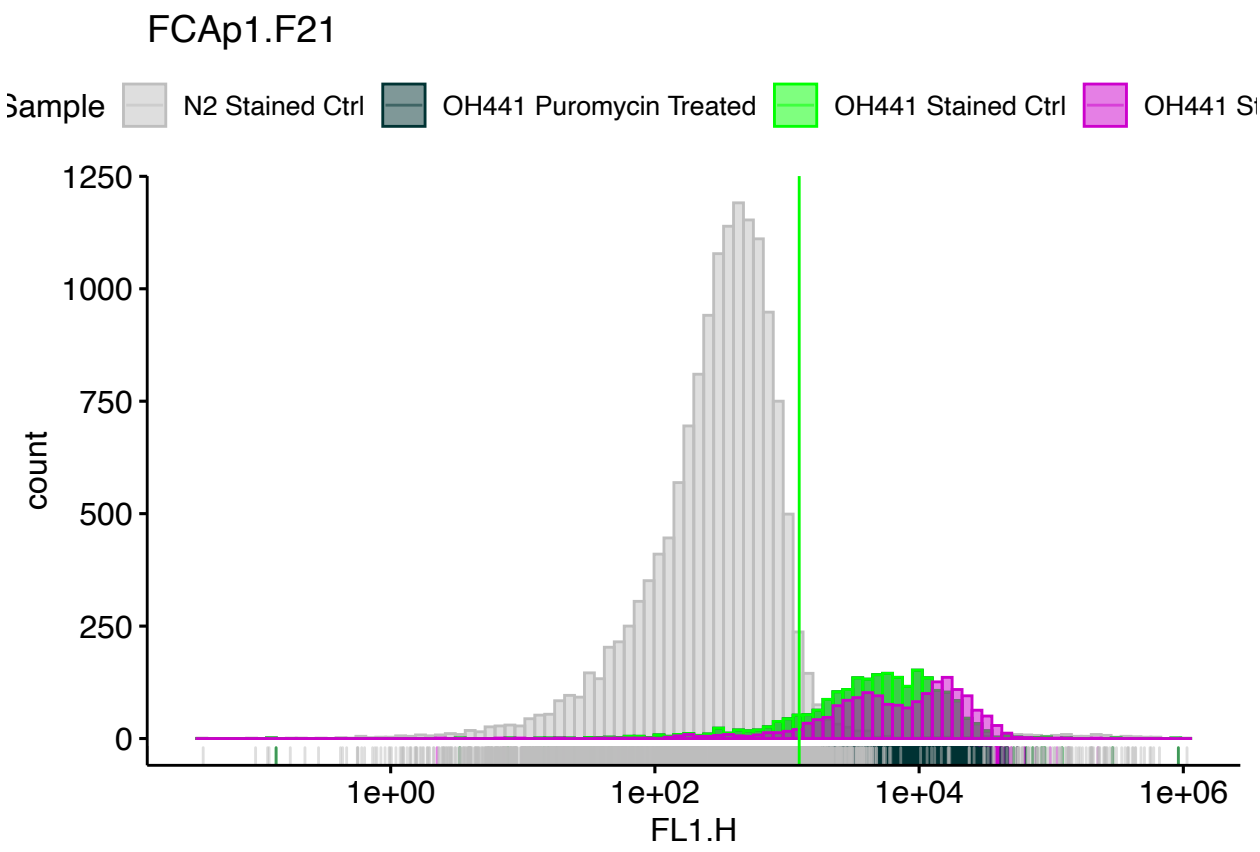
Transfer data to a dataframe to allow easy visual inspection

In this step we will also pool events from the respective .fcs files.

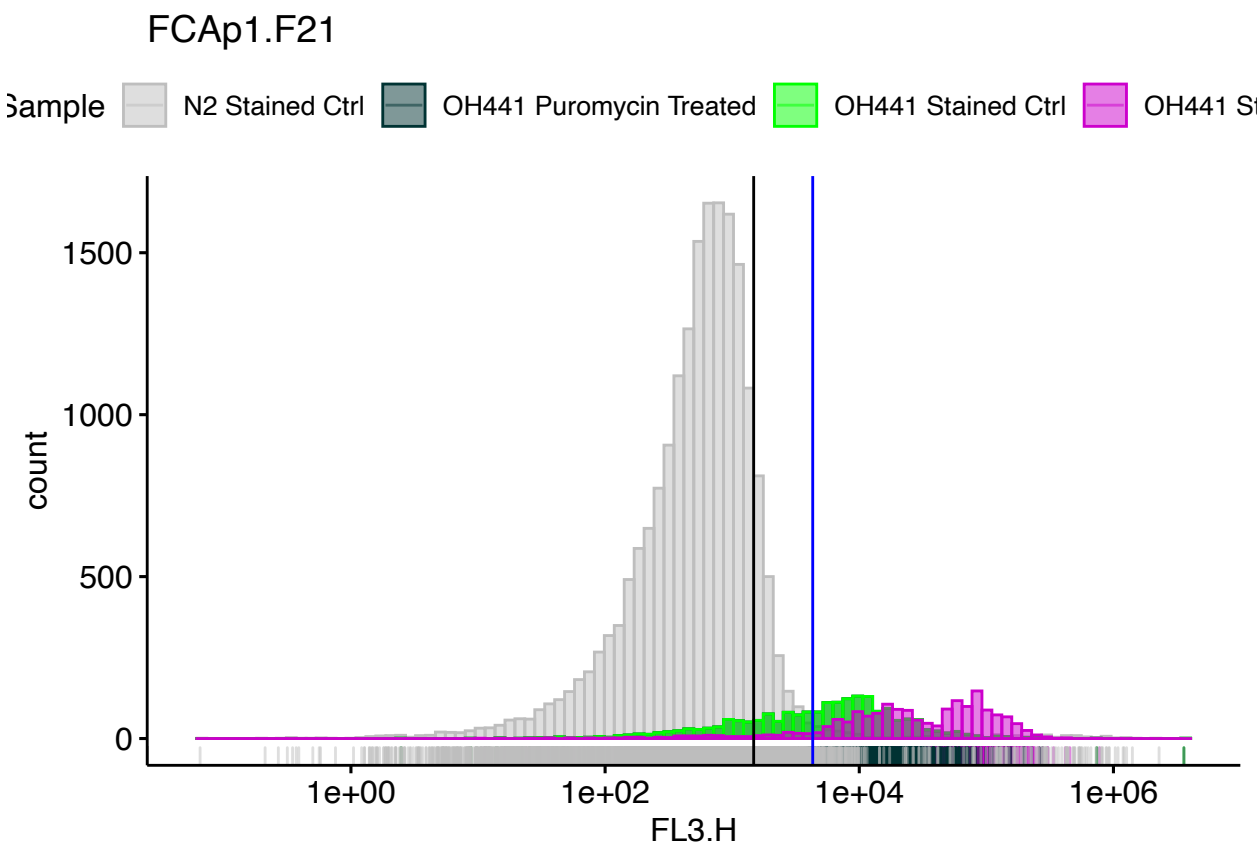
```
fl1h.N2.S.2 <- exprs(fsN2.S.2[[2]])[,"FL1-H"]
fl3h.N2.S.2 <- exprs(fsN2.S.2[[2]])[,"FL3-H"]
dfN2.S.2 <- data.frame("Sample"="N2 Stained Ctrl","FL1-H"=fl1h.N2.S.2,
                       "FL3-H"=fl3h.N2.S.2)

fl1h.SL.3 <- exprs(ffGFP.SL.3)[,"FL1-H"]
fl3h.SL.3 <- exprs(ffGFP.SL.3)[,"FL3-H"]
dfSL.3 <- data.frame("Sample"="OH441 Stained Ctrl","FL1-H"=fl1h.SL.3,
                       "FL3-H"=fl3h.SL.3)

fl1h.SD.3 <- exprs(ffGFP.SD.3)[,"FL1-H"]
fl3h.SD.3 <- exprs(ffGFP.SD.3)[,"FL3-H"]
dfSD.3 <- data.frame("Sample"="OH441 Stained HKC","FL1-H"=fl1h.SD.3,
                       "FL3-H"=fl3h.SD.3)

fl1h.SP.3 <- exprs(ffGFP.SP.3)[,"FL1-H"]
fl3h.SP.3 <- exprs(ffGFP.SP.3)[,"FL3-H"]
dfSP.3 <- data.frame("Sample"="OH441 Puromycin Treated","FL1-H"=fl1h.SP.3,
                       "FL3-H"=fl3h.SP.3)

dfAll.Comp <- rbind(dfSL.3,dfSD.3,dfSP.3,dfN2.S.2)
```

Visualize data

```
t <- "FCAp1.F21"
p <- gghistogram(dfAll.Comp, x = "FL1.H", rug = TRUE,
                       color = "Sample", fill = "Sample",
                       palette = c("grey","#003333","green","#CC00CC"),
                  bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfAll.Comp.FL1H.png"),p, scale =1, dpi=300)
p
```




```
t <- "FCAp1.F21"
p <- gghistogram(dfAll.Comp, x = "FL3.H", rug = TRUE,
                      color = "Sample", fill = "Sample",
                      palette = c("grey","#003333","green","#CC00CC"),
                  bins = 100)+
  scale_x_log10()+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  geom_vline(xintercept = ltU, color = "blue")+
  ggtitle(t)
ggsave(paste(dirSave,"1DHist.dfAll.Comp.FL3H.png"),p, scale =1, dpi=300)
p
```



Save R data to file

```
save.image(file = fnRdata )
```

A.2 Visualization of flow sets

This module is for visualizing and comparing multiple samples as flow sets and plots the classifer cutoffs as determined in module 1.

FCA_DissFig_Ch2_p2_Final

JFranco

4/8/2021

FCA Dissertation Figures for Chapter 2, part 2 - FINAL version

Part 2 of the analysis code is primarily for visualizing data with the previously derived cutoffs and for exporting data to .csv files (both actual flow cytometry data values as well as summary statistics)

Install the necessary packages

This installation procedure only needs to be conducted once. The code is left commented out as a helpful reminder if this is your first time setting up the R environment for FlowCytoAnalysis

```
#The code below should only need to be run once to install the necessary packages
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")

#BiocManager::install("flowCore")
#BiocManager::install("ggcyto")
#BiocManager::install("flowStats")
#install.packages("tidyverse")
#install.packages("scales")
#install.packages("ggpubr")
```

Load libraries

Although you do not need to install packages every time you run this code, you do need to load the libraries from said packages.

```
library(flowCore)
library(flowStats)
library(ggcyto)
library(ggplot2)
library(dplyr)
library(ggpubr)
require(scales)
require(gridExtra)
```

Load the environment from part 1

```
load("FCA_DissFig_Ch2_p1_Final_4to10.Rdata")
```

fs.to.df

This function copies values from a flowSet into a data frame, preserving unique sample and prep identifiers. There are limitations to visualizing flowSet data and these limitations are most easily overcome by generating plots from data frames rather than flowSets.

```
fs.to.df <- function(flowset){
  df.final <- data.frame()
  j = 1
  for (i in 1:length(flowset)){
    df.temp <- data.frame(
      filename = attr(flowset[[j]],"description")[[4]],
      FSCH = exprs(flowset[[j]][,"FSC-H"]),
      FSCA = exprs(flowset[[j]][,"FSC-A"]),
      SSCH = exprs(flowset[[j]][,"SSC-H"]),
      FL1H = exprs(flowset[[j]][,"FL1-H"]),
      FL3H = exprs(flowset[[j]][,"FL3-H"]),
      strain = pData(flowset)$strain[j],
      group = pData(flowset)$group[j]
      )
    df.final <- rbind(df.final, df.temp)
    j = j+1
  }
  df.final
}
```

Specify working directory and prep folder

This will be the main location for both getting raw data and storing analysis results.

```
folder <- "CCP_084/"
prep <- "CCP_084"
dirTA <- paste(dirData,folder,sep='')
```

Generate a flowset based on .fcs files in the prep folder

Reads in every .fcs file within the prep directory.

```
setwd(dirTA)
files <- list.files(pattern = "\\.fcs$")
fsTA <- read.flowSet(files, transformation=FALSE)
```

Create a directory for storing the results of the flow analysis

Analysis results will go into a subfolder within the main prep directory.

```
subDir.analysis <- "Analysis_p2_Final_4to10"
dir.create(file.path(dirTA,subDir.analysis))
```

Add a factor to the FlowSet to enable efficient plotting

These factors are used to help organize the data on the plots produced by ggplot. Every flowset is given the parameters "prep" and "sample." These parameters are easily transferred when the flowset is converted to a data frame and will make plotting easier.

```
j=1
for(i in files){
  pData(fsTA)$strain[j] <- substr(files[j],4,9)
  pData(fsTA)$group[j] <- substr(files[j],11,13)
  j= j+1
}
pData(fsTA)
```

```
##                                           name strain group
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100
```

Generate documentation for raw data

Convert the flowset to a data frame and save the raw data to .csv

The data frame will increase the number of options for visualizing the data

```
f <- paste(prep,"FS_Raw.csv",sep = ".")
dfTAraw <- fs.to.df(fsTA)
setwd(file.path(dirTA,subDir.analysis))
write.csv(dfTAraw, file = f, row.names = TRUE)
f
```

```
## [1] "CCP_084.FS_Raw.csv"
```

```
head(dfTAraw)
```

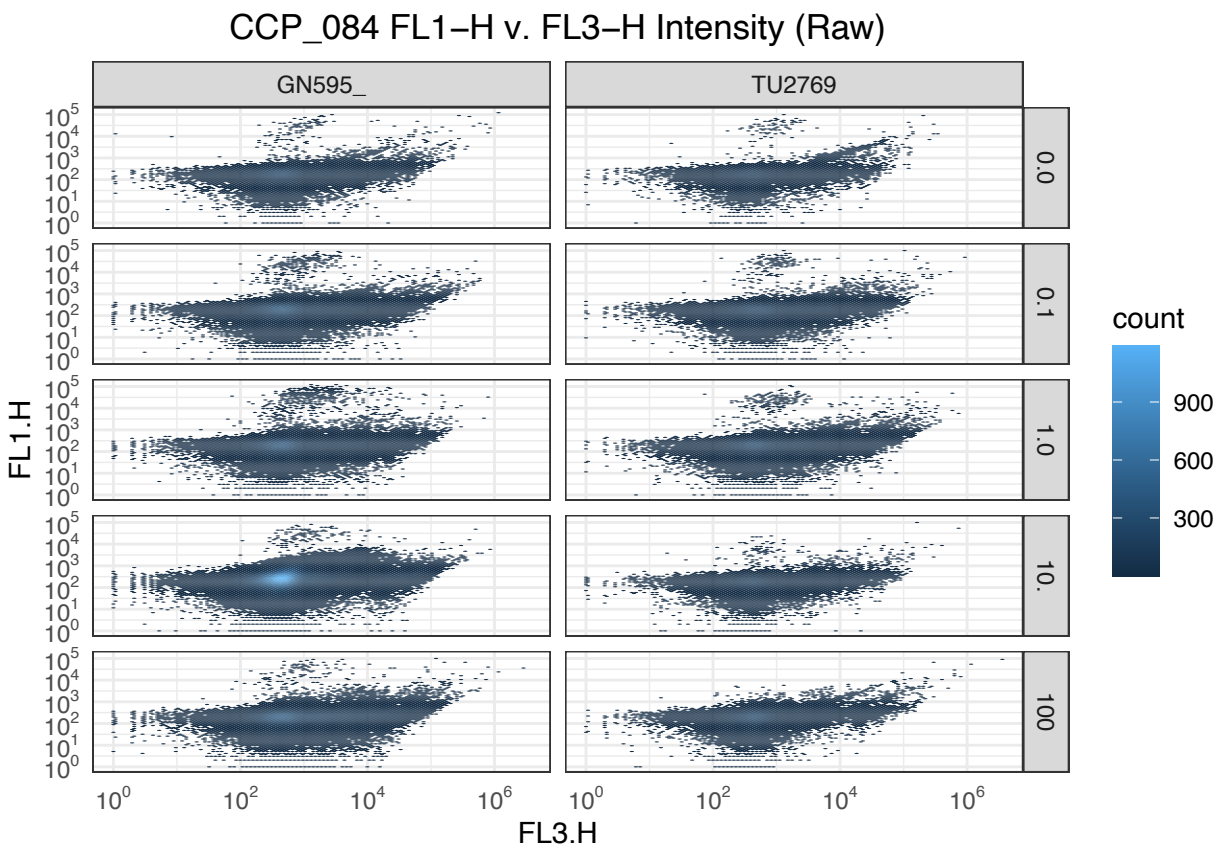
```
##           filename  FSC.H  FSC.A SSC.H FL1.H FL3.H strain group
## 1 GN595__000ugml-1 118314 131662 17182    72   597 GN595_   0.0
## 2 GN595__000ugml-1 103637  57683  6524   244   475 GN595_   0.0
## 3 GN595__000ugml-1 278521 165131  7114   204   195 GN595_   0.0
## 4 GN595__000ugml-1  81957  45147  5914    28   846 GN595_   0.0
## 5 GN595__000ugml-1  86043  46955 18435   191   151 GN595_   0.0
## 6 GN595__000ugml-1 257525 165499 34970    85  6039 GN595_   0.0
```

Visualize and save plots of the raw data

```
f <- paste(prep,"FL3H_FL1H_Raw.png",sep = ".")
p <- ggplot(dfTAraw, aes(x=FL3.H, y=FL1.H)) +
  geom_hex(bins=100) +
  facet_grid(group~strain) +
  scale_y_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  scale_x_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  theme_bw() +
  ggtitle(paste(prep," FL1-H v. FL3-H Intensity (Raw)",sep="")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
f
```

```
## [1] "CCP_084.FL3H_FL1H_Raw.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
## Saving 6.5 x 4.5 in image
```

Create a directory for storing Summary Statistics

```
subDir.ss <- "Summary_Stats"
dir.create(file.path(dirTA,subDir.analysis, subDir.ss))
```

Print summary statistics and save to CSV

Print total number of events

```
j = 1
for(i in files){
  print(paste(pData(fsTA)$name[j], " has ", nrow(fsTA[[j]]), " events.", sep = ""))
        j = j + 1
}
```

```
## [1] "01_GN595__0.0ugml-1.fcs has 40854 events."
## [1] "02_GN595__0.1ugml-1.fcs has 72821 events."
## [1] "03_GN595__1.0ugml-1.fcs has 74684 events."
## [1] "04_GN595__10.ugml-1.fcs has 199296 events."
## [1] "05_GN595__100ugml-1.fcs has 107136 events."
## [1] "06_TU2769_0.0ugml-1.fcs has 24841 events."
## [1] "07_TU2769_0.1ugml-1.fcs has 44774 events."
## [1] "08_TU2769_1.0ugml-1.fcs has 57177 events."
## [1] "09_TU2769_10.ugml-1.fcs has 30474 events."
## [1] "10_TU2769_100ugml-1.fcs has 44602 events."
```

Save summary statistics to CSV

```
setwd(file.path(dirTA, subDir.analysis, subDir.ss))
j = 1
for(i in files){
  f <- paste(pData(fsTA)$name[j],"SumStat_Raw.csv",sep = ".")
  df <- summary(fsTA)[[j]]
  write.csv(df, file = f, row.names = TRUE)
  j = j + 1
}
```

Gate data for debris

Debris is anything that's not the right size or is non-fluorescent. This step reuses the size and non-fluorescence gates defined in Part 1 and reloaded into the current environment. ### Apply the debris gates

```
fsTA.1 <- Subset(fsTA, gtSize)
```

Save gated data

```
f <- paste(prep,"FS_Gated.csv",sep = ".")
dfTAgated <- fs.to.df(fsTA.1)
setwd(file.path(dirTA,subDir.analysis))
write.csv(dfTAgated, file = f, row.names = TRUE)
f
```

```
## [1] "CCP_084.FS_Gated.csv"
```

Visualize and save plots of the debris gated data

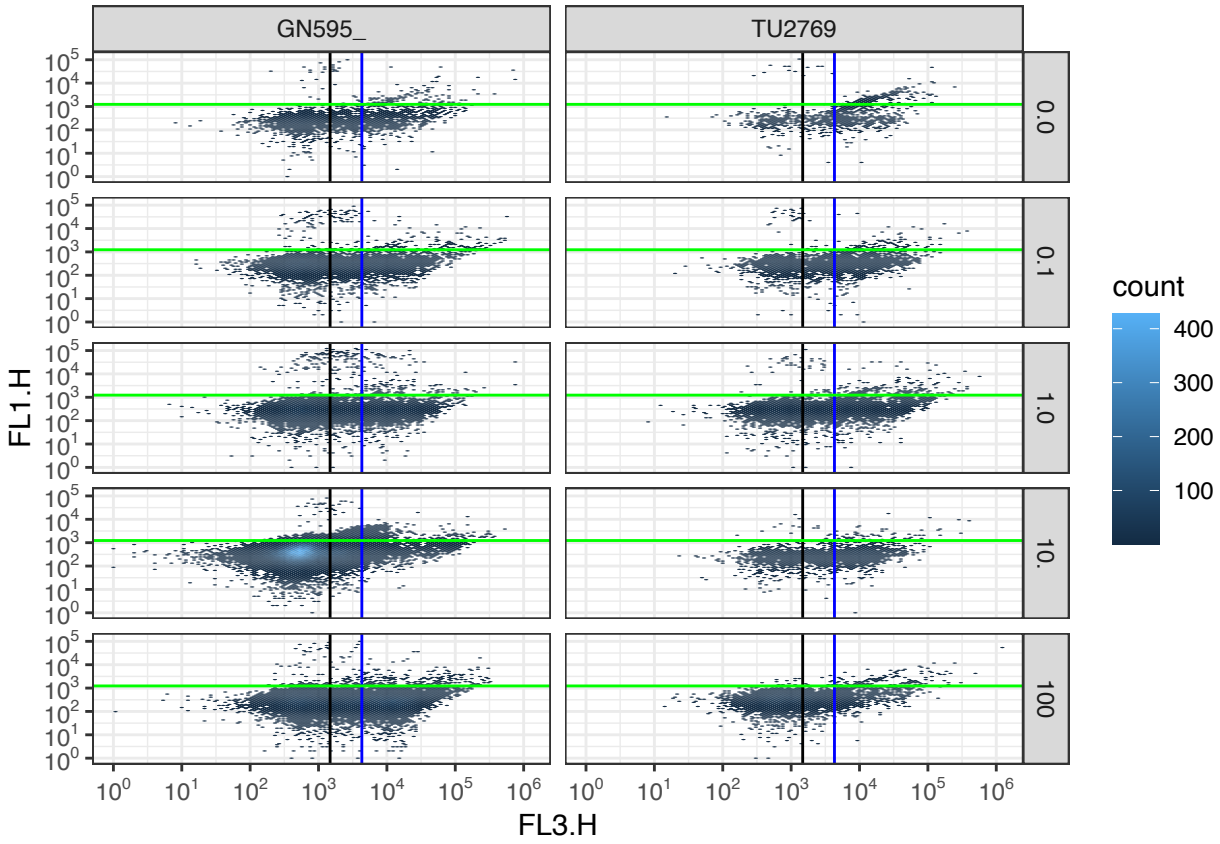
```
f <- paste(prep,"FL3H_FL1H_Gated.png",sep = ".")
p <- ggplot(dfTAgated, aes(x=FL3.H, y=FL1.H)) +
  geom_hex(bins=100) +
  facet_grid(group~strain) +
  scale_y_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  scale_x_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  theme_bw() +
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  geom_hline(yintercept = qGFP.US.2, color = "green")
  ggtitle(paste(prep," FL1-H v. FL3-H Intensity (Gated)",sep="")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
```

```
## NULL
```

```
f
```

```
## [1] "CCP_084.FL3H_FL1H_Gated.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
## Saving 6.5 x 4.5 in image
```

Print number of events remaining following gating

```
j = 1
for(i in files){
  print(paste(pData(fsTA.1)$name[j], " has ", nrow(fsTA.1[[j]]),
              " events.", sep = ""))
        j = j + 1
}
```

```
## [1] "01_GN595__0.0ugml-1.fcs has 2810 events."
## [1] "02_GN595__0.1ugml-1.fcs has 8493 events."
## [1] "03_GN595__1.0ugml-1.fcs has 10393 events."
## [1] "04_GN595__10.ugml-1.fcs has 57753 events."
## [1] "05_GN595__100ugml-1.fcs has 20117 events."
## [1] "06_TU2769_0.0ugml-1.fcs has 1147 events."
## [1] "07_TU2769_0.1ugml-1.fcs has 2799 events."
## [1] "08_TU2769_1.0ugml-1.fcs has 5516 events."
## [1] "09_TU2769_10.ugml-1.fcs has 2654 events."
## [1] "10_TU2769_100ugml-1.fcs has 6727 events."
```

Save summary statistics to file

```
setwd(file.path(dirTA, subDir.analysis, subDir.ss))
j = 1
for(i in files){
  f <- paste(pData(fsTA.1)$name[j],"SumStat_Gated.csv",sep = ".")
  df <- summary(fsTA.1)[[j]]
  write.csv(df, file = f, row.names = TRUE)
  j = j + 1
}
```

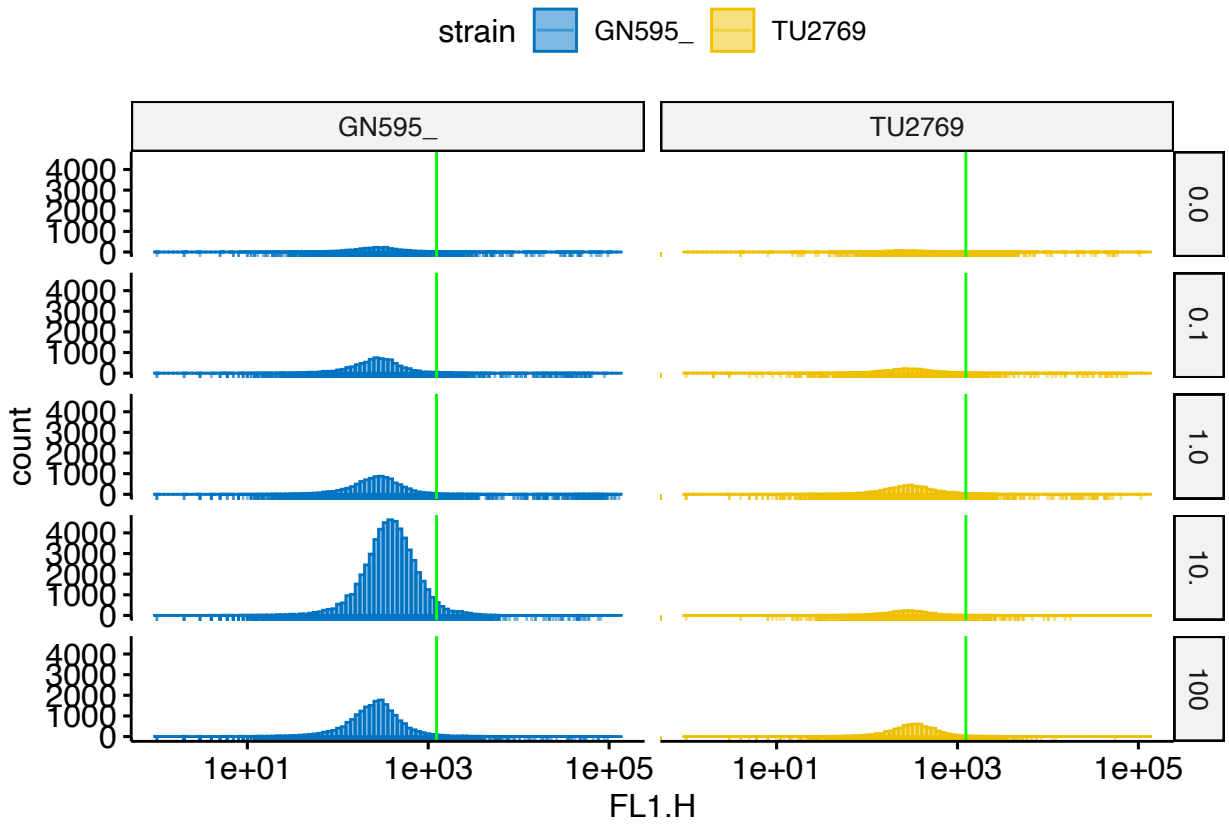
Visualize Densities in FL1-H and FL3-H

```
f <- paste(prep,"Hist.FL1H_Transd.png",sep = ".")
p <- gghistogram(dfTAgated, x = "FL1.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain)+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")
f
```

```
## [1] "CCP_084.Hist.FL1H_Transd.png"
```

```
p
```

```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 966 rows containing non-finite values (stat_bin).
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Removed 966 rows containing non-finite values (stat_bin).
```

```
f <- paste(prep,"Hist.FL3H_Transd.png",sep = ".")
p <- gghistogram(dfTAgated, x = "FL3.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain)+
  scale_x_log10()+
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = dtL, color = "red")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")
f
```

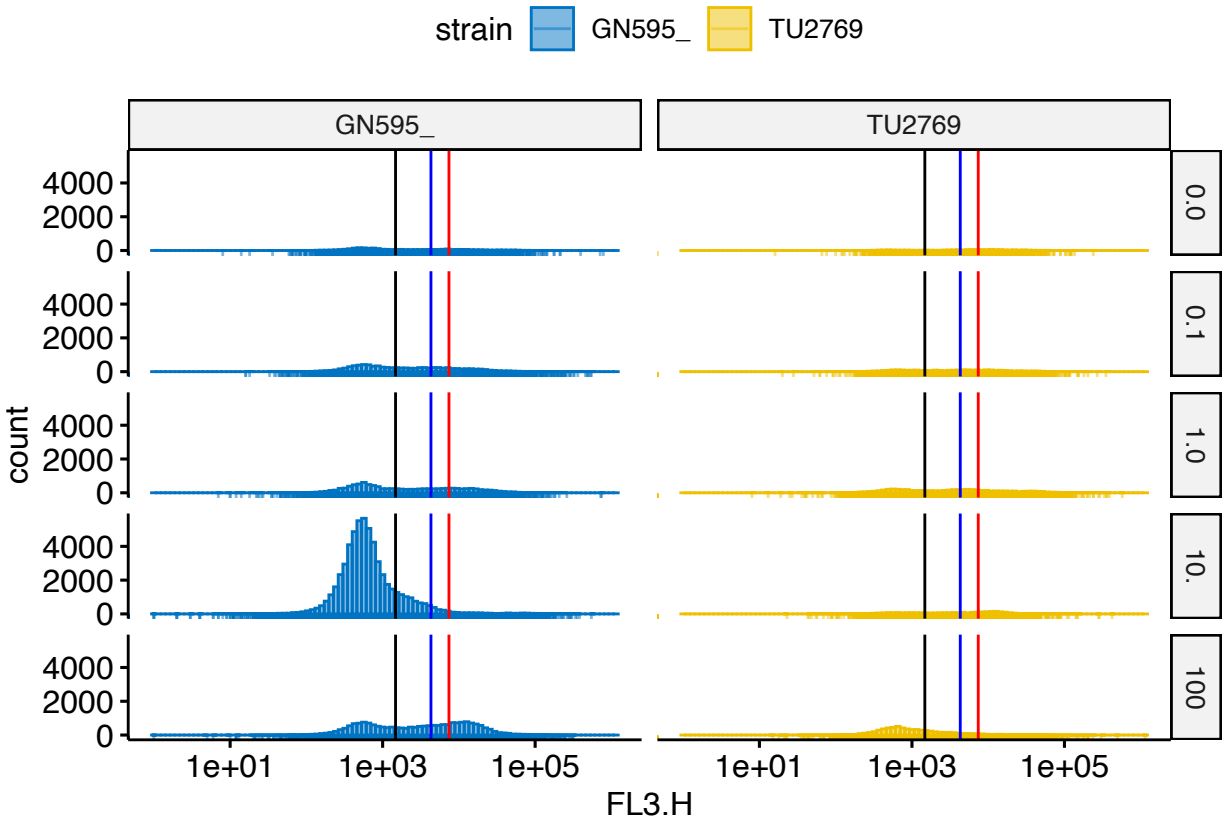
```
## [1] "CCP_084.Hist.FL3H_Transd.png"
```

```
p
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Removed 209 rows containing non-finite values (stat_bin).
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Transformation introduced infinite values in continuous x-axis
```

```
## Warning: Removed 209 rows containing non-finite values (stat_bin).
```

A.3 Quantification of cell types

This module quantifies the number of live, dead, and GFP positive or negative cells in the analyzed populations and generates plots to visualize these data.

FCA_DissFig_Ch2_p3

JFranco

4/8/2021

FCA Dissertation Figures for Chapter 2, part 3 - FINAL version

Part 3 of the analysis is for quantifying GFP(+) events and survival rates.

Install the necessary packages

This installation procedure only needs to be conducted once. The code is left commented out as a helpful reminder if this is your first time setting up the R environment for FlowCytoAnalysis

```
#The code below should only need to be run once to install the necessary packages
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")

#BiocManager::install("flowCore")
#BiocManager::install("ggcyto")
#BiocManager::install("flowStats")
#install.packages("tidyverse")
#install.packages("scales")
#install.packages("ggpubr")
```

Load libraries

Although you do not need to install packages every time you run this code, you do need to load the libraries from said packages.

```
library(flowCore)
library(flowStats)
library(ggcyto)
library(ggplot2)
library(dplyr)
library(ggpubr)
require(scales)
require(gridExtra)
```

Load the environment from part 1

Nothing from p2 analysis is necessary since p2 is more of a quality control step.

```
load("FCA_DissFig_Ch2_p1_Final_4to10.Rdata")
```

fs.to.df

This function copies values from a flowSet into a data frame, preserving unique sample and prep identifiers. There are limitations to visualizing flowSet data and these limitations are most easily overcome by generating plots from data frames rather than flowSets.

```
fs.to.df <- function(flowset){
  df.final <- data.frame()
  j = 1
  for (i in 1:length(flowset)){
    df.temp <- data.frame(
      filename = attr(flowset[[j]],"description")[[4]],
      FSCH = exprs(flowset[[j]][,"FSC-H"]),
      FSCA = exprs(flowset[[j]][,"FSC-A"]),
      SSCH = exprs(flowset[[j]][,"SSC-H"]),
      FL1H = exprs(flowset[[j]][,"FL1-H"]),
      FL3H = exprs(flowset[[j]][,"FL3-H"]),
      strain = pData(flowset)$strain[j],
      group = pData(flowset)$group[j]
      )
    df.final <- rbind(df.final, df.temp)
    j = j+1
  }
  df.final
}
```

Specify working directory and prep folder

This will be the main location for both getting raw data and storing analysis results.

```
folder <- "CCP_084/"
prep <- "CCP_084"
dirTA <- paste(dirData,folder,sep='')
```

Generate a flowset based on .fcs files in the prep folder

Reads in every .fcs file within the prep directory.

```
setwd(dirTA)
files <- list.files(pattern = "\\.fcs$")
fsTA <- read.flowSet(files, transformation=FALSE)
```

Create a directory for storing the results of the flow analysis

Analysis results will go into a subfolder within the main prep directory.

```
subDir.analysis <- "Analysis_p3_4to10"
dir.create(file.path(dirTA,subDir.analysis))
```

Add a factor to the FlowSet to enable efficient plotting

These factors are used to help organize the data on the plots produced by ggplot. Every flowset is given the parameters "prep" and "sample." These parameters are easily transferred when the flowset is converted to a data frame and will make plotting easier.

```
j=1
for(i in files){
  pData(fsTA)$strain[j] <- substr(files[j],4,9)
  pData(fsTA)$group[j] <- substr(files[j],11,13)
  j= j+1
}
pData(fsTA)
```

```
##                                              name strain group
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100
```

Gate data for debris and autofluorescence

Debris is anything that's not the right size or is non-fluorescent. This step reuses the size and non-fluorescence gates defined in Part 1 and reloaded into the current environment. ### Apply the debris gates This data is no different than what was saved in FCA Part 2.

```
fsTA.1 <- Subset(fsTA, gtSize)
```

Apply autofluorescence gates generating two populations

```
fsTA.2.AF <- Subset(fsTA.1, gtAF)
fsTA.2.NAF <- Subset(fsTA.1, gtNAF)
```

Convert the flowset to a data frame and save the raw data to .csv

The data frame will increase the number of options for visualizing the data

```
f <- paste(prep,"FS_AF.csv",sep = ".")
dfTA.2.AF <- fs.to.df(fsTA.2.AF)
setwd(file.path(dirTA,subDir.analysis))
write.csv(dfTA.2.AF, file = f, row.names = TRUE)
f
```

```
## [1] "CCP_084.FS_AF.csv"
```

```
head(dfTA.2.AF)
```

```
##           filename  FSC.H  FSC.A  SSC.H FL1.H FL3.H strain group
## 1 GN595__000ugml-1 509735 323836  34787   133   403 GN595_   0.0
## 2 GN595__000ugml-1 475460 360848  22382   211  2347 GN595_   0.0
## 3 GN595__000ugml-1 459134 274844  20733     0   440 GN595_   0.0
## 4 GN595__000ugml-1 611167 516179 243227   552   715 GN595_   0.0
## 5 GN595__000ugml-1 483359 293096  60445   264   780 GN595_   0.0
## 6 GN595__000ugml-1 762563 482196  27112   134   772 GN595_   0.0
```

```
f <- paste(prep,"FS_NAF.csv",sep = ".")
dfTA.2.NAF <- fs.to.df(fsTA.2.NAF)
setwd(file.path(dirTA,subDir.analysis))
write.csv(dfTA.2.NAF, file = f, row.names = TRUE)
f
```

```
## [1] "CCP_084.FS_NAF.csv"
```

```
head(dfTA.2.NAF)
```

```
##           filename  FSC.H  FSC.A  SSC.H FL1.H FL3.H strain group
## 1 GN595__000ugml-1 913817 626680  76262   346   350 GN595_   0.0
## 2 GN595__000ugml-1 611167 516179 243227   552   715 GN595_   0.0
## 3 GN595__000ugml-1 450666 264834  15283   361   493 GN595_   0.0
## 4 GN595__000ugml-1 694602 499621  40711   251   479 GN595_   0.0
## 5 GN595__000ugml-1 696037 432362  24000   307   810 GN595_   0.0
## 6 GN595__000ugml-1 468504 286181  28053    87   601 GN595_   0.0
```

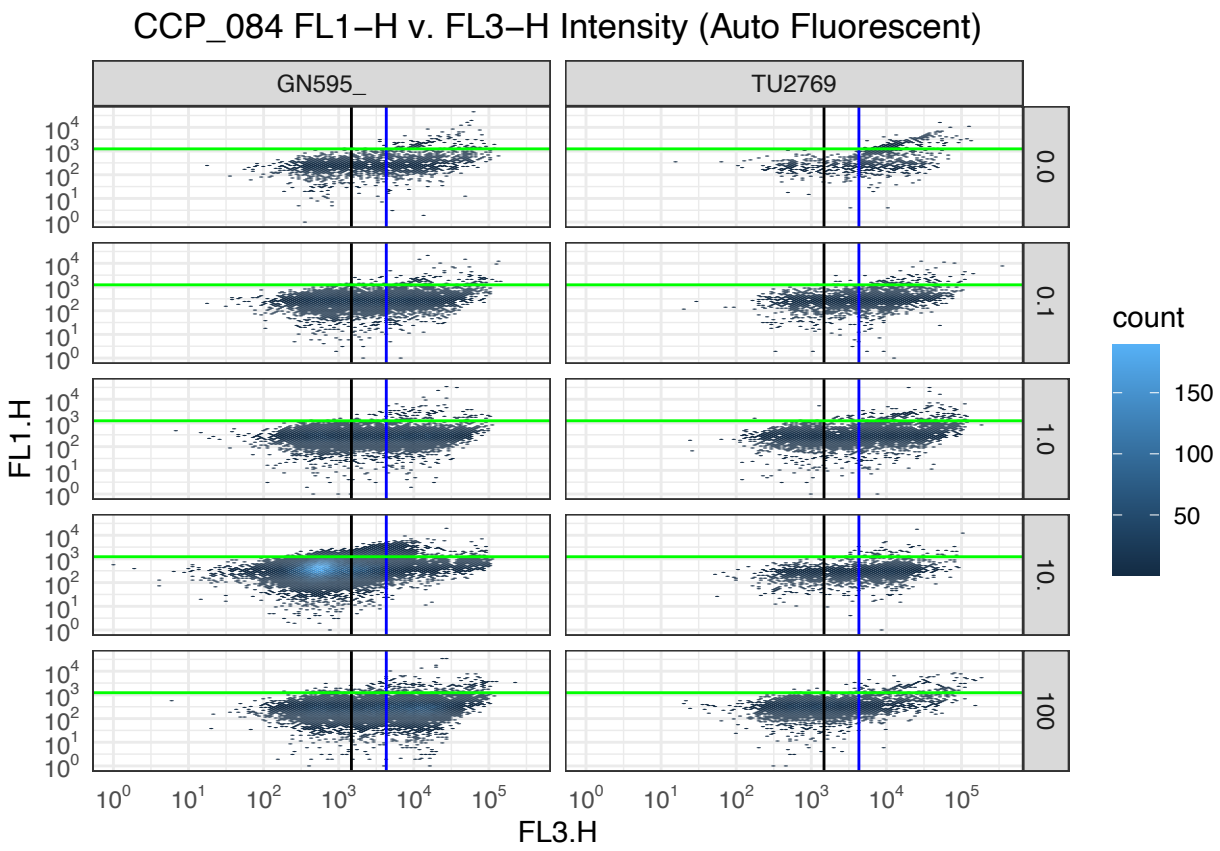
Visualize gated autofluorescent population

```
f <- paste(prep,"FL3H_FL1H_AF.png",sep = ".")
p <- ggplot(dfTA.2.AF, aes(x=FL3.H, y=FL1.H)) +
  geom_hex(bins=100) +
  facet_grid(group~strain) +
  scale_y_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  scale_x_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  theme_bw() +
```

```
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  geom_hline(yintercept = qGFP.US.2, color = "green")+
  ggtitle(paste(prep," FL1-H v. FL3-H Intensity (Auto Fluorescent)",sep="")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
f
```

```
## [1] "CCP_084.FL3H_FL1H_AF.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

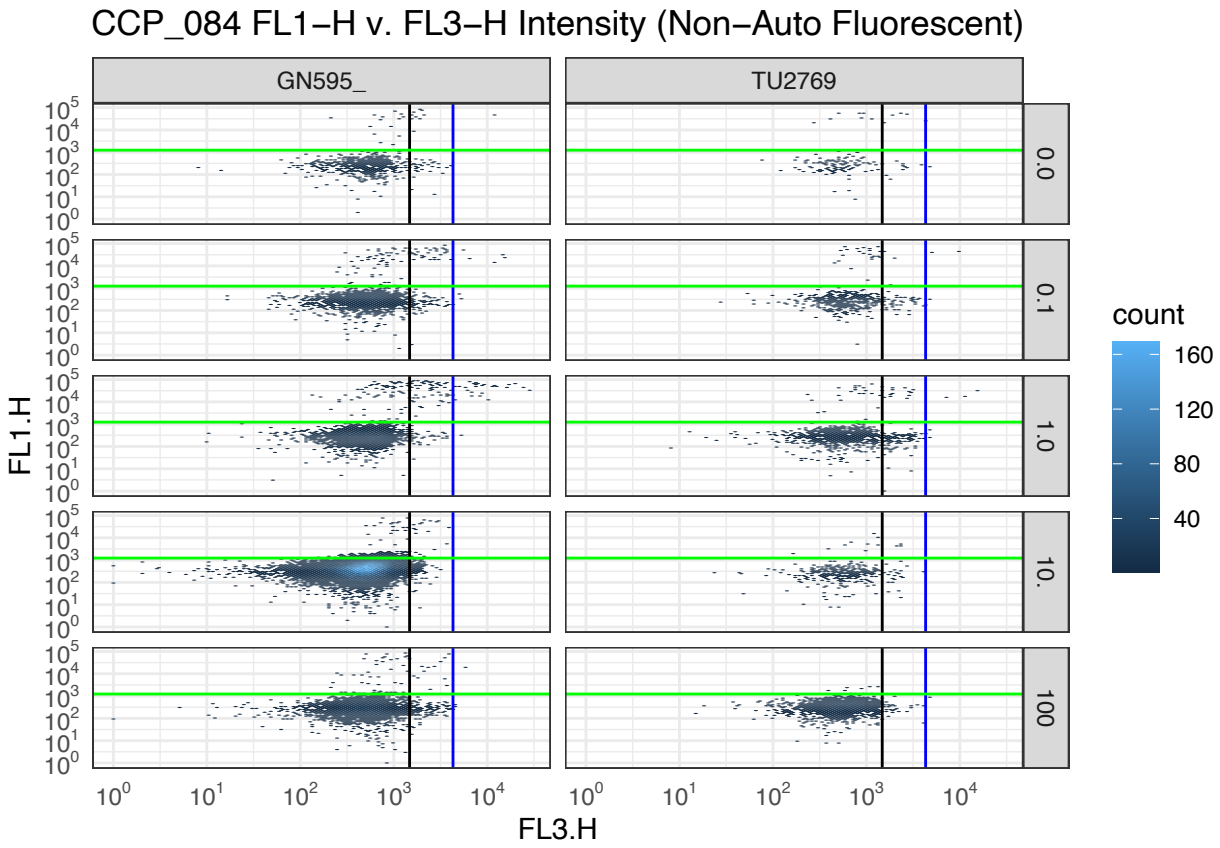
Visualize gated non-autofluorescent population

```
f <- paste(prep,"FL3H_FL1H_NAF.png",sep = ".")
p <- ggplot(dfTA.2.NAF, aes(x=FL3.H, y=FL1.H)) +
  geom_hex(bins=100) +
  facet_grid(group~strain) +
  scale_y_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
```

```
  scale_x_continuous(trans = log10_trans(),
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x))) +
  theme_bw() +
  ggtitle(paste(prep," FL1-H v. FL3-H Intensity (Non-Auto Fluorescent)",sep="")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())+
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  geom_hline(yintercept = qGFP.US.2, color = "green")
f
```

```
## [1] "CCP_084.FL3H_FL1H_NAF.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

Print number of events per sample per subset

```
print("Auto-fluorescent group")
```

```
## [1] "Auto-fluorescent group"
```

```
j = 1
for(i in files){
  print(paste(pData(fsTA.2.AF)$name[j], " has ", nrow(fsTA.2.AF[[j]]),
              " events.", sep = ""))
        j = j + 1
}
```

```
## [1] "01_GN595__0.0ugml-1.fcs has 2125 events."
## [1] "02_GN595__0.1ugml-1.fcs has 6671 events."
## [1] "03_GN595__1.0ugml-1.fcs has 8125 events."
## [1] "04_GN595__10.ugml-1.fcs has 37470 events."
## [1] "05_GN595__100ugml-1.fcs has 17022 events."
## [1] "06_TU2769_0.0ugml-1.fcs has 1015 events."
## [1] "07_TU2769_0.1ugml-1.fcs has 2359 events."
## [1] "08_TU2769_1.0ugml-1.fcs has 4563 events."
## [1] "09_TU2769_10.ugml-1.fcs has 2314 events."
## [1] "10_TU2769_100ugml-1.fcs has 5022 events."
```

```
print("Non-Auto fluorescent group")
```

```
## [1] "Non-Auto fluorescent group"
```

```
j = 1
for(i in files){
  print(paste(pData(fsTA.2.NAF)$name[j], " has ", nrow(fsTA.2.NAF[[j]]),
              " events.", sep = ""))
        j = j + 1
}
```

```
## [1] "01_GN595__0.0ugml-1.fcs has 738 events."
## [1] "02_GN595__0.1ugml-1.fcs has 1981 events."
## [1] "03_GN595__1.0ugml-1.fcs has 2529 events."
## [1] "04_GN595__10.ugml-1.fcs has 24164 events."
## [1] "05_GN595__100ugml-1.fcs has 3462 events."
## [1] "06_TU2769_0.0ugml-1.fcs has 141 events."
## [1] "07_TU2769_0.1ugml-1.fcs has 492 events."
## [1] "08_TU2769_1.0ugml-1.fcs has 1039 events."
## [1] "09_TU2769_10.ugml-1.fcs has 364 events."
## [1] "10_TU2769_100ugml-1.fcs has 1972 events."
```

Save summary statistics to file

```
setwd(file.path(dirTA, subDir.analysis))
j = 1
for(i in files){
  f <- paste(pData(fsTA.2.AF)$name[j],"SumStat_AF.csv",sep = ".")
  df <- summary(fsTA.2.AF)[[j]]
  write.csv(df, file = f, row.names = TRUE)
  j = j + 1
}
```

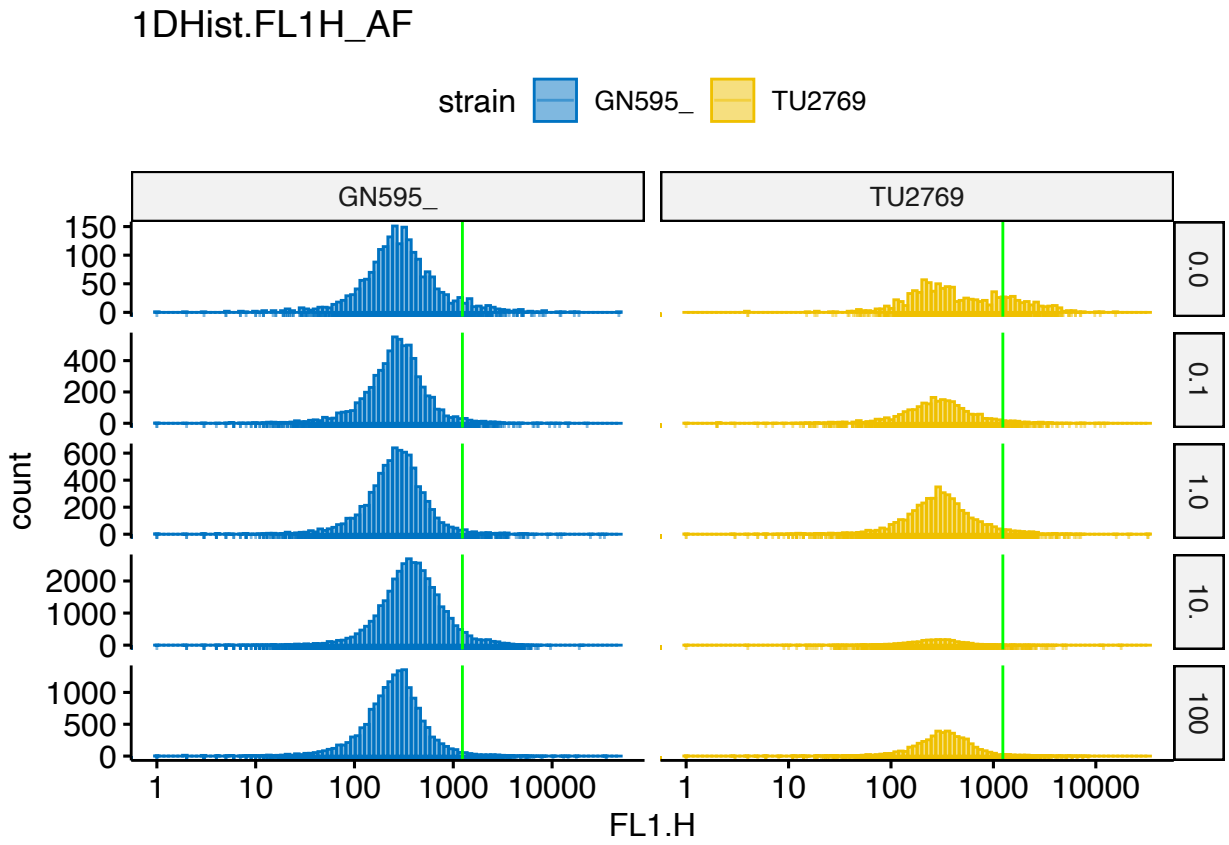
```
j = 1
for(i in files){
  f <- paste(pData(fsTA.2.NAF)$name[j],"SumStat_NAF.csv",sep = ".")
  df <- summary(fsTA.2.NAF)[[j]]
  write.csv(df, file = f, row.names = TRUE)
  j = j + 1
}
```

Visualize Densities in FL1-H and FL3-H For Autofluorescent

```
t <- "1DHist.FL1H_AF"
f <- paste(prep,t,".png",sep = ".")
p <- gghistogram(dfTA.2.AF, x = "FL1.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain,scales="free")+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
f
```

```
## [1] "CCP_084.1DHist.FL1H_AF..png"
```

```
p
```

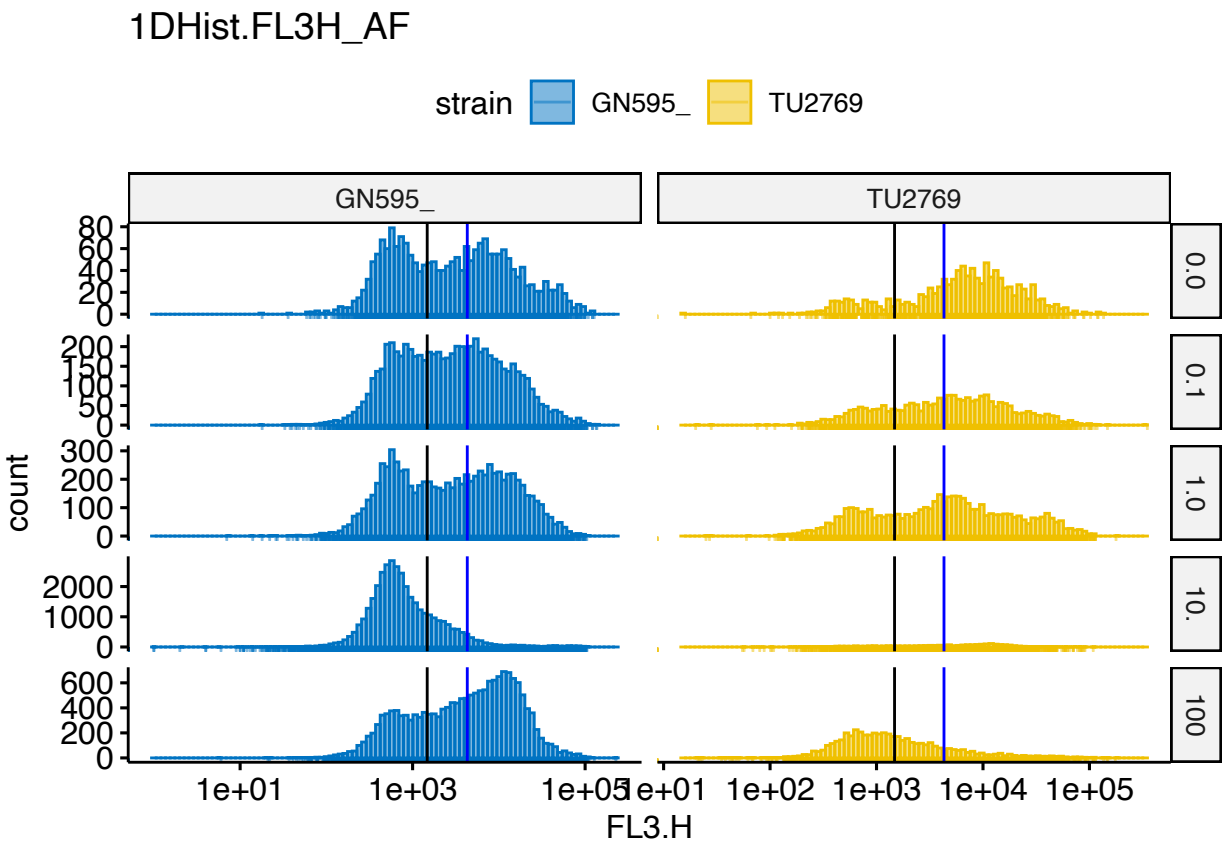


```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
t <- "1DHist.FL3H_AF"
f <- paste(prep,t,".png",sep = ".")
p <- gghistogram(dfTA.2.AF, x = "FL3.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain,scales="free")+
  scale_x_log10()+
  geom_vline(xintercept = ltU, color = "blue")+
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
f
```

```
## [1] "CCP_084.1DHist.FL3H_AF..png"
```

```
p
```



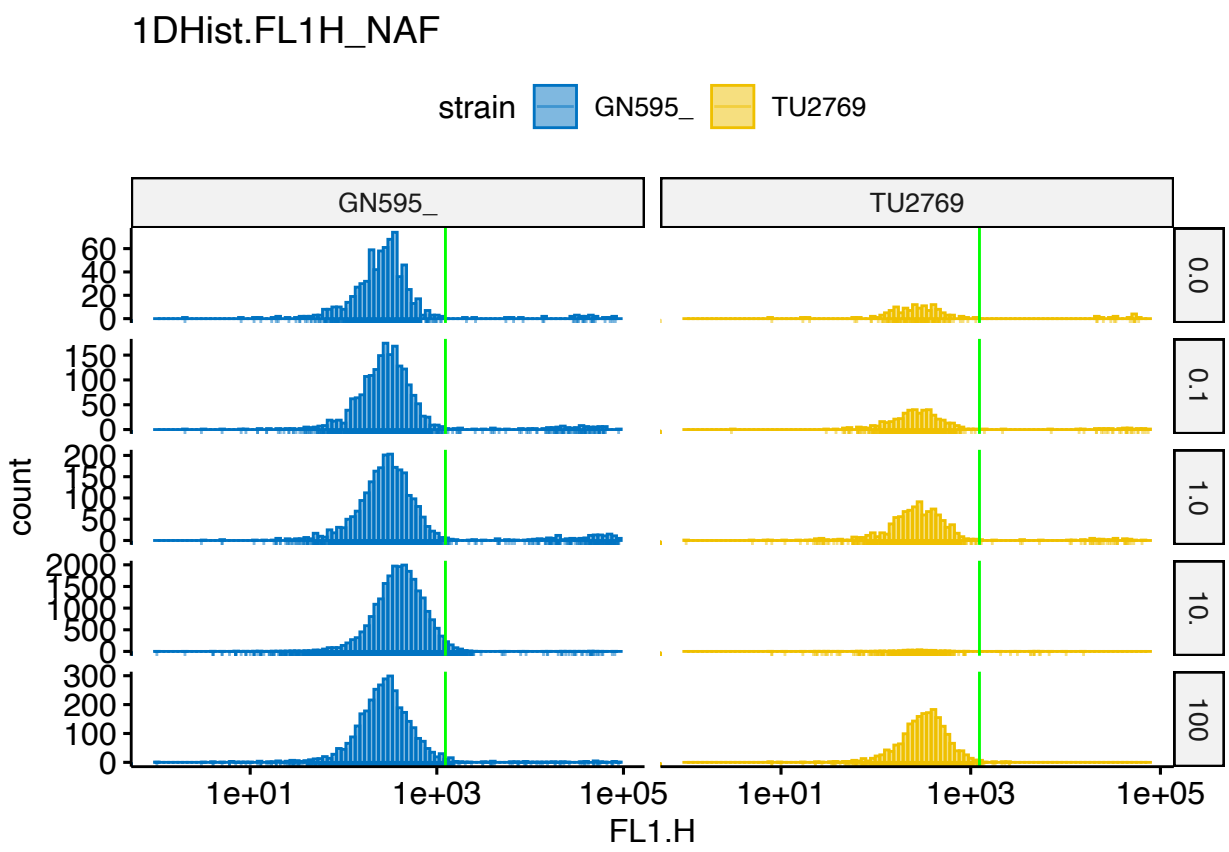

```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

Visualize Densities in FL1-H and FL3-H For Non-autofluorescent

```
t <- "1DHist.FL1H_NAF"
f <- paste(prep,t,".png",sep = ".")
p <- gghistogram(dfTA.2.NAF, x = "FL1.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain,scales="free")+
  scale_x_log10()+
  geom_vline(xintercept = qGFP.US.2, color = "green")+
  ggtitle(t)
f
```

```
## [1] "CCP_084.1DHist.FL1H_NAF..png"
```

```
p
```



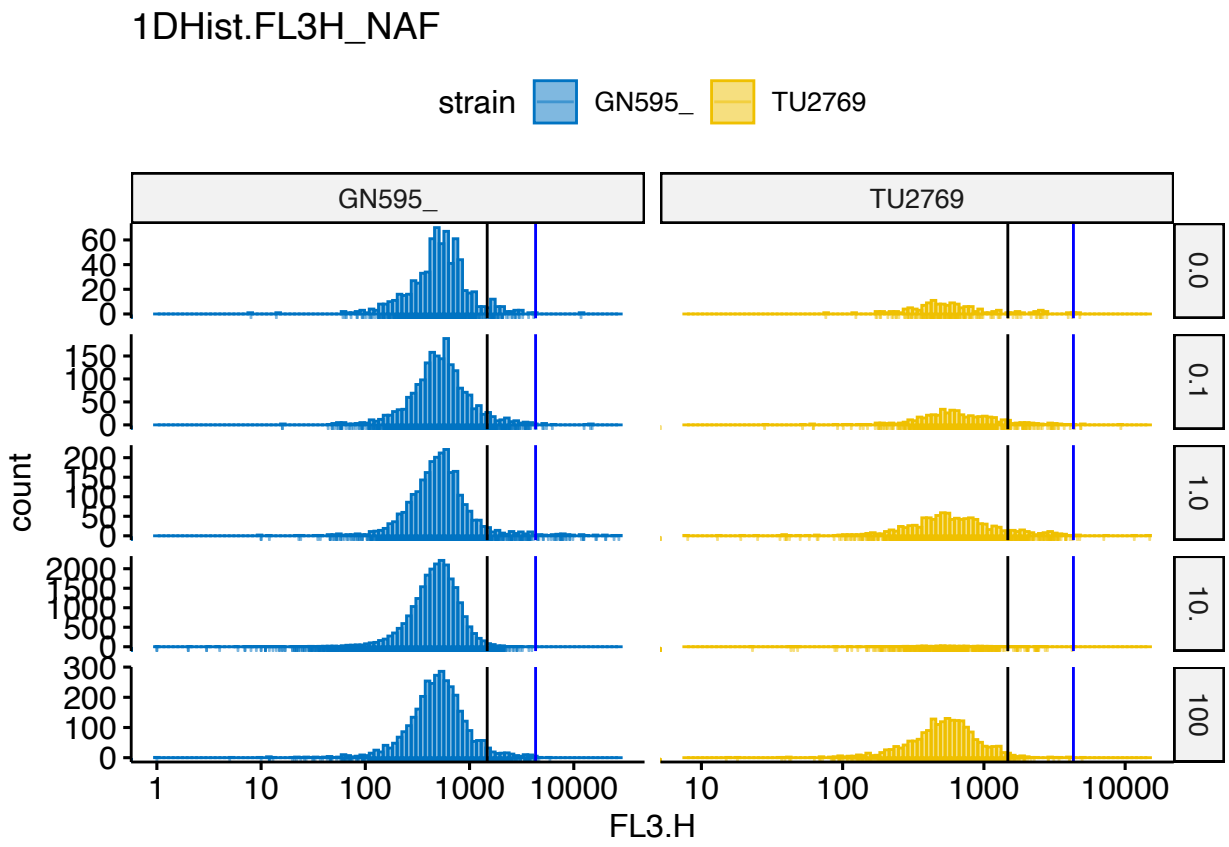

```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

```
t <- "1DHist.FL3H_NAF"
f <- paste(prep,t,".png",sep = ".")
p <- gghistogram(dfTA.2.NAF, x = "FL3.H", rug = TRUE,
                 color = "strain", fill = "strain",
                     palette = "jco", bins = 100)+
  facet_grid(group~strain,scales="free")+
  scale_x_log10()+
  geom_vline(xintercept = ltU, color = "blue")+
```

```
  geom_vline(xintercept = qD.N2.PR.S.1, color = "black")+
  ggtitle(t)
f
```

```
## [1] "CCP_084.1DHist.FL3H_NAF..png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",f,sep=""), p, dpi = 300)
```

Define gates for classifying live/dead, GFP/non-GFP

```
# Live gate is used for AF
gate.live <- rectangleGate(filterId = "All live cells",
                           "FL3-H"=c(0,qD.N2.PR.S.1))
gate.gfp.live <- rectangleGate(filterId = "GFP Positive & Live",
                               "FL1-H"=c(qGFP.US.2,Inf),
                               "FL3-H"=c(0,qD.N2.PR.S.1))
gate.gfp.all <- rectangleGate(filterId = "All GFP(+) cells", "FL1-H"=c(qGFP.US.2,Inf))
```

Store TRN ratio as pData value for each flowframe for AF

Need to process both AF and NF in order to be able to accurately count GFP samples. Dead non-GFP cells look exactly like live GFP cells in 2D (FL1-FL3), but can be detected in (FL1-FL2) analysis. Breaking the samples into AF and NAF allows us to gate in 3D. Cells above the GFP line in this group are most likely

```
j=1
for(i in files){
  # Store number of events in self (for checking)
  events <- nrow(fsTA.2.AF[[j]])
  pData(fsTA.2.AF)$events[j] <- events

  # Classify events as live/dead
  result.live = flowCore::filter(fsTA.2.AF[[j]],gate.live)
  # Count the number of events classified as live
  cells.live = as.numeric(summary(result.live)$true)

  # Store live/dead values
  pData(fsTA.2.AF)$live[j] <- cells.live
  pData(fsTA.2.AF)$dead[j] <- events - cells.live
  j= j+1
}
pData(fsTA.2.AF)
```

```
##                                            name strain group events  live  dead
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0   2125   820  1305
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1   6671  2433  4238
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0   8125  3111  5014
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.  37470 29094  8376
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100 17022  4620 12402
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0   1015   165   850
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1   2359   558  1801
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0   4563  1278  3285
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.   2314   511  1803
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100   5022  2885  2137
```

Store TRN ratio as pData value for each flowframe for NAF

Need to process both AF and NF in order to be able to accurately count GFP samples. Dead non-GFP cells look exactly like live GFP cells in 2D (FL1-FL3), but can be detected in (FL1-FL2) analysis. Breaking the samples into AF and NAF allows us to gate in 3D

```
j=1
for(i in files){
  # Store number of events in self
  events <- nrow(fsTA.2.NAF[[j]])
  pData(fsTA.2.NAF)$events[j] <- events

  # Classify events as live/dead
  result.live = flowCore::filter(fsTA.2.NAF[[j]],gate.live)
  # Count number of live cells
  cells.live = as.numeric(summary(result.live)$true)

  pData(fsTA.2.NAF)$live[j] <- cells.live
  pData(fsTA.2.NAF)$dead[j] <- events - cells.live

  # Classify events as GFP or not
  result.gfp = flowCore::filter(fsTA.2.NAF[[j]],gate.gfp.all)
```

```
  cells.gfp = as.numeric(summary(result.gfp)$true)
  pData(fsTA.2.NAF)$gfp[j] <- cells.gfp

  # Classify events as GFP and live
  result.gfp.live = flowCore::filter(fsTA.2.NAF[[j]],gate.gfp.live)
  # Count number of true events
  cells.gfp.live = as.numeric(summary(result.gfp.live)$true)

  # Store values
  pData(fsTA.2.NAF)$gfp.live[j] <- cells.gfp.live
  pData(fsTA.2.NAF)$gfp.dead[j] <- cells.gfp - cells.gfp.live

  j= j+1
}
pData(fsTA.2.NAF)
```

```
##                                          name strain group events  live dead
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0    738   697   41
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1   1981  1866  115
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0   2529  2400  129
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.  24164 23987  177
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100   3462  3324  138
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0    141   125   16
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1    492   445   47
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0   1039   920  119
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.    364   339   25
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100   1972  1935   37
##                         gfp gfp.live gfp.dead
## 01_GN595__0.0ugml-1.fcs  25       14       11
## 02_GN595__0.1ugml-1.fcs  92       54       38
## 03_GN595__1.0ugml-1.fcs 164       75       89
## 04_GN595__10.ugml-1.fcs 512      492       20
## 05_GN595__100ugml-1.fcs  76       62       14
## 06_TU2769_0.0ugml-1.fcs  12        7        5
## 07_TU2769_0.1ugml-1.fcs  27       22        5
## 08_TU2769_1.0ugml-1.fcs  40       25       15
## 09_TU2769_10.ugml-1.fcs  13       10        3
## 10_TU2769_100ugml-1.fcs  11       11        0
```

Store the pData in a DF

This will make plotting easier

```
df.pData.final.AF <- pData(fsTA.2.AF)
df.pData.final.NAF <- pData(fsTA.2.NAF)
df.pData.final.All <- pData(fsTA.1)
```

Calculate survival and gfp percentages

```
j=1
s2marker <- (length(files)/2) + 1
for(i in files){

  # Get values
  events.all <- nrow(fsTA.1[[j]])         # Total number of events pre AF gating
  live.af <- pData(fsTA.2.AF)$live[j]     # Live events in AF group
  dead.af <- pData(fsTA.2.AF)$dead[j]     # Dead events in AF group
  live.naf <- pData(fsTA.2.NAF)$live[j]   # Live events in NAF group
  dead.naf <- pData(fsTA.2.NAF)$dead[j]   # Dead events in NAF group
  live.all <- live.af+live.naf            # All live
  dead.all <- dead.af+dead.naf            # All dead
  live.gfp <- pData(fsTA.2.NAF)$gfp.live[j]     # Live and GFP events
  dead.gfp <- pData(fsTA.2.NAF)$gfp.dead[j]     # Dead and GFP events
  gfp.all <- live.gfp+dead.gfp            # All GFP events

  # Calculate survival rates
        # What percent of cells survive treatment
  survival.all <- 100*live.all/(live.all+dead.all)
        # What percent of GFP(+) survive treatment
  survival.gfp <- 100*live.gfp/(gfp.all)

  # Calculate GFP percentage
      # What percent of living cells are GFP?
  perc.gfp.live <- 100*(live.gfp/live.all)

  # Store values
  pData(fsTA.1)$events[j] <- events.all
  pData(fsTA.1)$events.gfp[j] <- gfp.all

  pData(fsTA.1)$live.af[j] <- live.af
  pData(fsTA.1)$dead.af[j] <- dead.af

  pData(fsTA.1)$live.naf[j] <- live.naf
  pData(fsTA.1)$dead.naf[j] <- dead.naf

  pData(fsTA.1)$live.all[j] <- live.all
  pData(fsTA.1)$dead.all[j] <- dead.all

  pData(fsTA.1)$live.gfp[j] <- live.gfp
  pData(fsTA.1)$dead.gfp[j] <- dead.gfp

  pData(fsTA.1)$survival.all[j] <- survival.all
  pData(fsTA.1)$survival.gfp[j] <- survival.gfp

  pData(fsTA.1)$perc.gfp.live[j] <- perc.gfp.live

  # Store ctrl data for calculating enrichment later
  if(j==1){
    perc.gfp.s1 <- perc.gfp.live
    s1 <- pData(fsTA.1)$strain[j]
    g1 <- pData(fsTA.1)$group[j]
    }
```

```
  if(j==(s2marker)){
    perc.gfp.s2 <- perc.gfp.live
    s2 <- pData(fsTA.1)$strain[j]
    g2 <- pData(fsTA.1)$group[j]
    }

  j= j+1
}
```

Calculate fold change

```
j=1
for(i in files){
  if(j < s2marker){
    pData(fsTA.1)$foldchange[j] <- pData(fsTA.1)$perc.gfp.live[j]/perc.gfp.s1
  }
  if(j >= s2marker){
    pData(fsTA.1)$foldchange[j] <- pData(fsTA.1)$perc.gfp.live[j]/perc.gfp.s2
  }
  j=j+1
}
pData(fsTA.1)
```

```
##                                          name strain group events events.gfp
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0   2810         25
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1   8493         92
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0  10393        164
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.  57753        512
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100  20117         76
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0   1147         12
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1   2799         27
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0   5516         40
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.   2654         13
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100   6727         11
##                         live.af dead.af live.naf dead.naf live.all dead.all
## 01_GN595__0.0ugml-1.fcs     820    1305      697       41     1517     1346
## 02_GN595__0.1ugml-1.fcs    2433    4238     1866      115     4299     4353
## 03_GN595__1.0ugml-1.fcs    3111    5014     2400      129     5511     5143
## 04_GN595__10.ugml-1.fcs   29094    8376    23987      177    53081     8553
## 05_GN595__100ugml-1.fcs    4620   12402     3324      138     7944    12540
## 06_TU2769_0.0ugml-1.fcs     165     850      125       16      290      866
## 07_TU2769_0.1ugml-1.fcs     558    1801      445       47     1003     1848
## 08_TU2769_1.0ugml-1.fcs    1278    3285      920      119     2198     3404
## 09_TU2769_10.ugml-1.fcs     511    1803      339       25      850     1828
## 10_TU2769_100ugml-1.fcs    2885    2137     1935       37     4820     2174
##                         live.gfp dead.gfp survival.all survival.gfp
## 01_GN595__0.0ugml-1.fcs       14       11     52.98638     56.00000
## 02_GN595__0.1ugml-1.fcs       54       38     49.68793     58.69565
## 03_GN595__1.0ugml-1.fcs       75       89     51.72705     45.73171
## 04_GN595__10.ugml-1.fcs      492       20     86.12292     96.09375
## 05_GN595__100ugml-1.fcs       62       14     38.78149     81.57895
```

```
## 06_TU2769_0.0ugml-1.fcs        7        5     25.08651     58.33333
## 07_TU2769_0.1ugml-1.fcs       22        5     35.18064     81.48148
## 08_TU2769_1.0ugml-1.fcs       25       15     39.23599     62.50000
## 09_TU2769_10.ugml-1.fcs       10        3     31.74010     76.92308
## 10_TU2769_100ugml-1.fcs       11        0     68.91621    100.00000
##                          perc.gfp.live foldchange
## 01_GN595__0.0ugml-1.fcs     0.9228741 1.00000000
## 02_GN595__0.1ugml-1.fcs     1.2561061 1.36108065
## 03_GN595__1.0ugml-1.fcs     1.3609145 1.47464811
## 04_GN595__10.ugml-1.fcs     0.9268853 1.00434646
## 05_GN595__100ugml-1.fcs     0.7804632 0.84568767
## 06_TU2769_0.0ugml-1.fcs     2.4137931 1.00000000
## 07_TU2769_0.1ugml-1.fcs     2.1934197 0.90870246
## 08_TU2769_1.0ugml-1.fcs     1.1373976 0.47120759
## 09_TU2769_10.ugml-1.fcs     1.1764706 0.48739496
## 10_TU2769_100ugml-1.fcs     0.2282158 0.09454653
```

Store the pData in a DF

This will make plotting easier

```
df.pData.final <- pData(fsTA.1)
```

Setup plotting parameters

```
color.codes<-as.character(c("#0073C2FF", "#EFC000FF"))
strains<-unique(df.pData.final$strain)
```

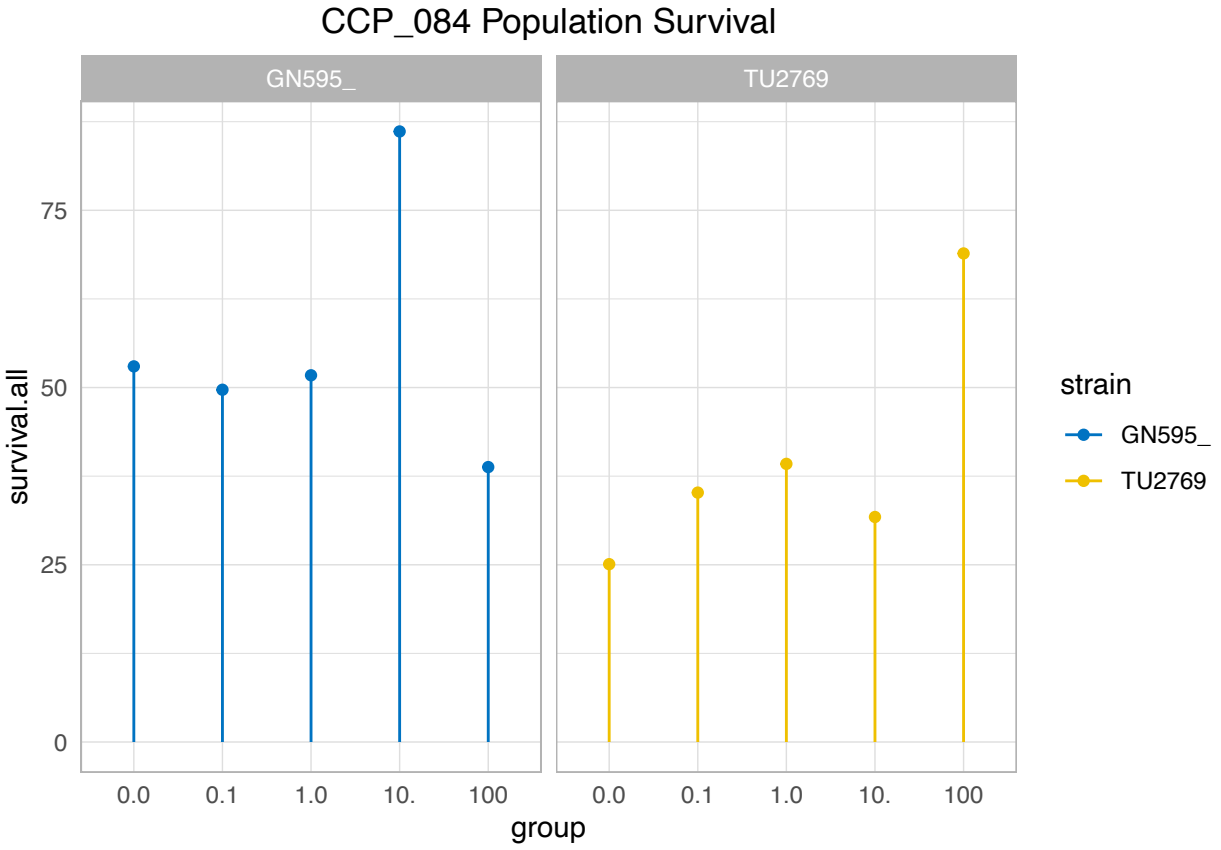
Plot what percent of all events are live

I.e., "Survival All"

```
filename <- paste(prep,"Survival_All.png",sep = ".")
p <- ggplot(df.pData.final, aes(x=group, y=survival.all,color=strain)) +
  #geom_bar(stat="identity", position=position_dodge()) +
  geom_point()+
  geom_segment(aes(x=group,xend=group,y=0,yend=survival.all))+
  theme_light()+
  scale_colour_manual(values=setNames(color.codes, strains))+
  facet_grid(~strain) +
  ggtitle(paste(prep,"Population Survival",sep=" ")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
filename
```

```
## [1] "CCP_084.Survival_All.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",filename,sep=""), p, dpi = 300)
```

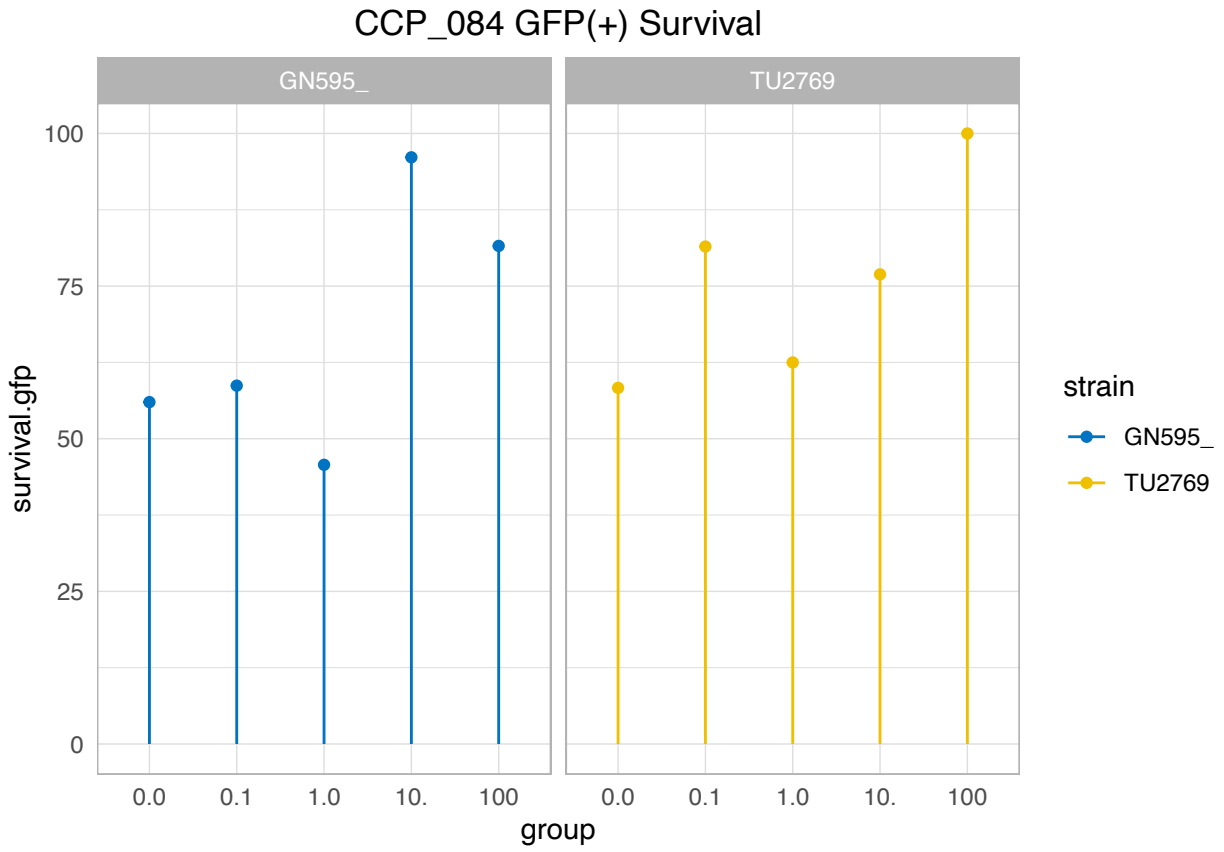
Plot what percent of all GFP are live

I.e., "Survival GFP"

```
filename <- paste(prep,"Survival_GFP.png",sep = ".")
p <- ggplot(df.pData.final, aes(x=group, y=survival.gfp,color=strain)) +
  #geom_bar(stat="identity", position=position_dodge()) +
  geom_point()+
  geom_segment(aes(x=group,xend=group,y=0,yend=survival.gfp))+
  theme_light()+
  scale_colour_manual(values=setNames(color.codes, strains))+
  facet_grid(~strain) +
  ggtitle(paste(prep,"GFP(+) Survival",sep=" ")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
filename
```

```
## [1] "CCP_084.Survival_GFP.png"
```

```
p
```



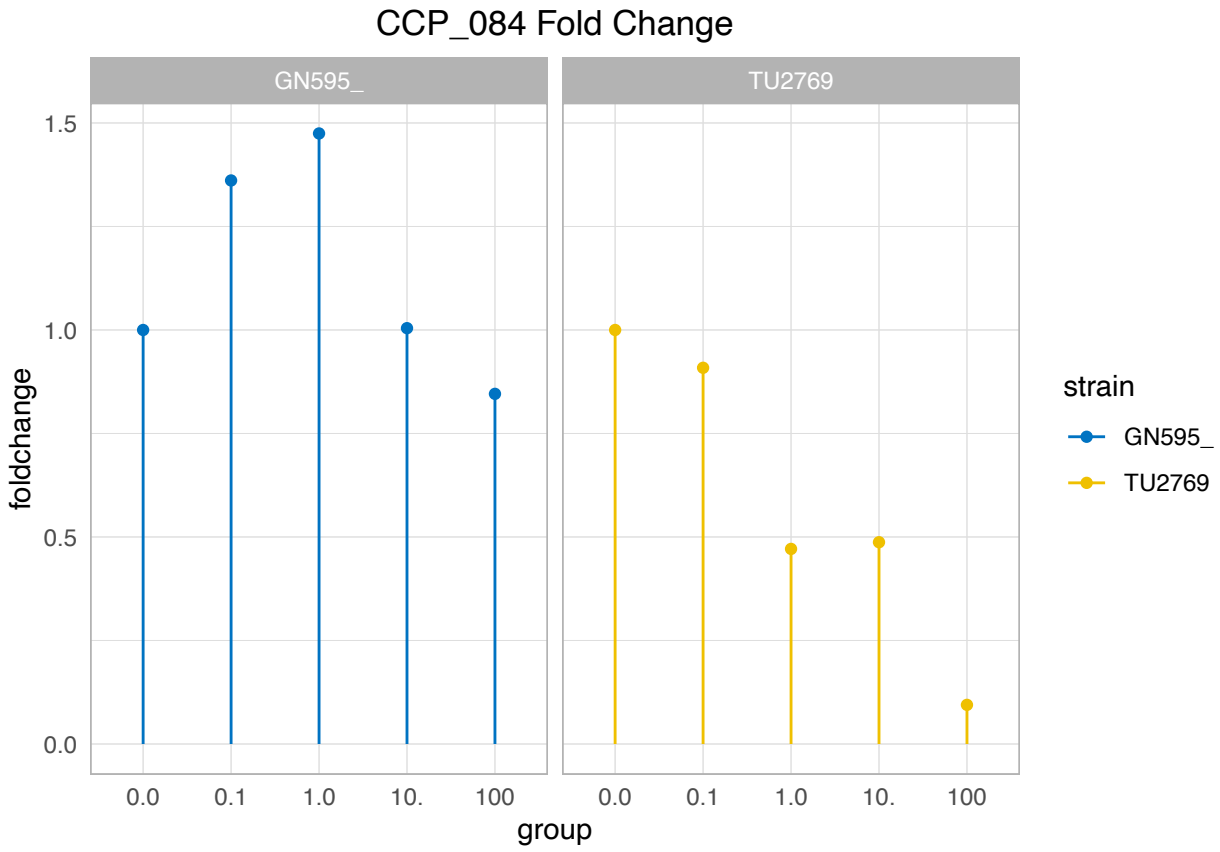

```
ggsave(paste(dirTA,subDir.analysis,"/",filename,sep=""), p, dpi = 300)
```

Plot Fold Change

```
filename <- paste(prep,"FoldChange.png",sep = ".")

p <- ggplot(df.pData.final, aes(x=group, y=foldchange,color=strain)) +
  #geom_bar(stat="identity", position=position_dodge()) +
  geom_point()+
  geom_segment(aes(x=group,xend=group,y=0,yend=foldchange))+
  theme_light()+
  scale_colour_manual(values=setNames(color.codes, strains))+
  facet_grid(~strain) +
  ggtitle(paste(prep,"Fold Change",sep=" ")) +
  theme(plot.title=element_text(hjust = 0.5),
        axis.ticks = element_blank())
filename
```

```
## [1] "CCP_084.FoldChange.png"
```

```
p
```




```
ggsave(paste(dirTA,subDir.analysis,"/",filename,sep=""), p, dpi = 300)
```

Save values to a CSV file and print

```
setwd(file.path(dirTA,subDir.analysis))

filename <- paste("Summary_Final.csv")
write.csv(df.pData.final, file = filename, row.names = TRUE)

df.pData.final
```

```
##                                          name strain group events events.gfp
## 01_GN595__0.0ugml-1.fcs 01_GN595__0.0ugml-1.fcs GN595_   0.0   2810         25
## 02_GN595__0.1ugml-1.fcs 02_GN595__0.1ugml-1.fcs GN595_   0.1   8493         92
## 03_GN595__1.0ugml-1.fcs 03_GN595__1.0ugml-1.fcs GN595_   1.0  10393        164
## 04_GN595__10.ugml-1.fcs 04_GN595__10.ugml-1.fcs GN595_   10.  57753        512
## 05_GN595__100ugml-1.fcs 05_GN595__100ugml-1.fcs GN595_   100  20117         76
## 06_TU2769_0.0ugml-1.fcs 06_TU2769_0.0ugml-1.fcs TU2769   0.0   1147         12
## 07_TU2769_0.1ugml-1.fcs 07_TU2769_0.1ugml-1.fcs TU2769   0.1   2799         27
## 08_TU2769_1.0ugml-1.fcs 08_TU2769_1.0ugml-1.fcs TU2769   1.0   5516         40
## 09_TU2769_10.ugml-1.fcs 09_TU2769_10.ugml-1.fcs TU2769   10.   2654         13
## 10_TU2769_100ugml-1.fcs 10_TU2769_100ugml-1.fcs TU2769   100   6727         11
##                         live.af dead.af live.naf dead.naf live.all dead.all
## 01_GN595__0.0ugml-1.fcs     820    1305      697       41     1517     1346
```

```
## 02_GN595__0.1ugml-1.fcs    2433    4238     1866      115     4299     4353
## 03_GN595__1.0ugml-1.fcs    3111    5014     2400      129     5511     5143
## 04_GN595__10.ugml-1.fcs   29094    8376    23987      177    53081     8553
## 05_GN595__100ugml-1.fcs    4620   12402     3324      138     7944    12540
## 06_TU2769_0.0ugml-1.fcs     165     850      125       16      290      866
## 07_TU2769_0.1ugml-1.fcs     558    1801      445       47     1003     1848
## 08_TU2769_1.0ugml-1.fcs    1278    3285      920      119     2198     3404
## 09_TU2769_10.ugml-1.fcs     511    1803      339       25      850     1828
## 10_TU2769_100ugml-1.fcs    2885    2137     1935       37     4820     2174
##                         live.gfp dead.gfp survival.all survival.gfp
## 01_GN595__0.0ugml-1.fcs       14       11     52.98638     56.00000
## 02_GN595__0.1ugml-1.fcs       54       38     49.68793     58.69565
## 03_GN595__1.0ugml-1.fcs       75       89     51.72705     45.73171
## 04_GN595__10.ugml-1.fcs      492       20     86.12292     96.09375
## 05_GN595__100ugml-1.fcs       62       14     38.78149     81.57895
## 06_TU2769_0.0ugml-1.fcs        7        5     25.08651     58.33333
## 07_TU2769_0.1ugml-1.fcs       22        5     35.18064     81.48148
## 08_TU2769_1.0ugml-1.fcs       25       15     39.23599     62.50000
## 09_TU2769_10.ugml-1.fcs       10        3     31.74010     76.92308
## 10_TU2769_100ugml-1.fcs       11        0     68.91621    100.00000
##                         perc.gfp.live foldchange
## 01_GN595__0.0ugml-1.fcs     0.9228741 1.00000000
## 02_GN595__0.1ugml-1.fcs     1.2561061 1.36108065
## 03_GN595__1.0ugml-1.fcs     1.3609145 1.47464811
## 04_GN595__10.ugml-1.fcs     0.9268853 1.00434646
## 05_GN595__100ugml-1.fcs     0.7804632 0.84568767
## 06_TU2769_0.0ugml-1.fcs     2.4137931 1.00000000
## 07_TU2769_0.1ugml-1.fcs     2.1934197 0.90870246
## 08_TU2769_1.0ugml-1.fcs     1.1373976 0.47120759
## 09_TU2769_10.ugml-1.fcs     1.1764706 0.48739496
## 10_TU2769_100ugml-1.fcs     0.2282158 0.09454653
```

Appendix B

ImageJ based analysis tools for cropping and tracing

B.1 Analysis details

The goal of the code described here is to generate a set of images that can be used for quantification of MEC-4 puncta localization (Appendix C) and colocalization with RAB-3 (Appendix D). To support the overarching aims of this thesis with regards to obtaining robust morphological measurements, these tools are designed to minimize operator input and thus measurement bias and error. These sets of code are written in the ImageJ macro language Jython, which is a hybrid of Java and Python. The .ijm files will be made available on GitHub with the publication of the manuscript described in Chapter 4. In this section I provide a graphical overview of the image analysis process and expected outputs from the code.

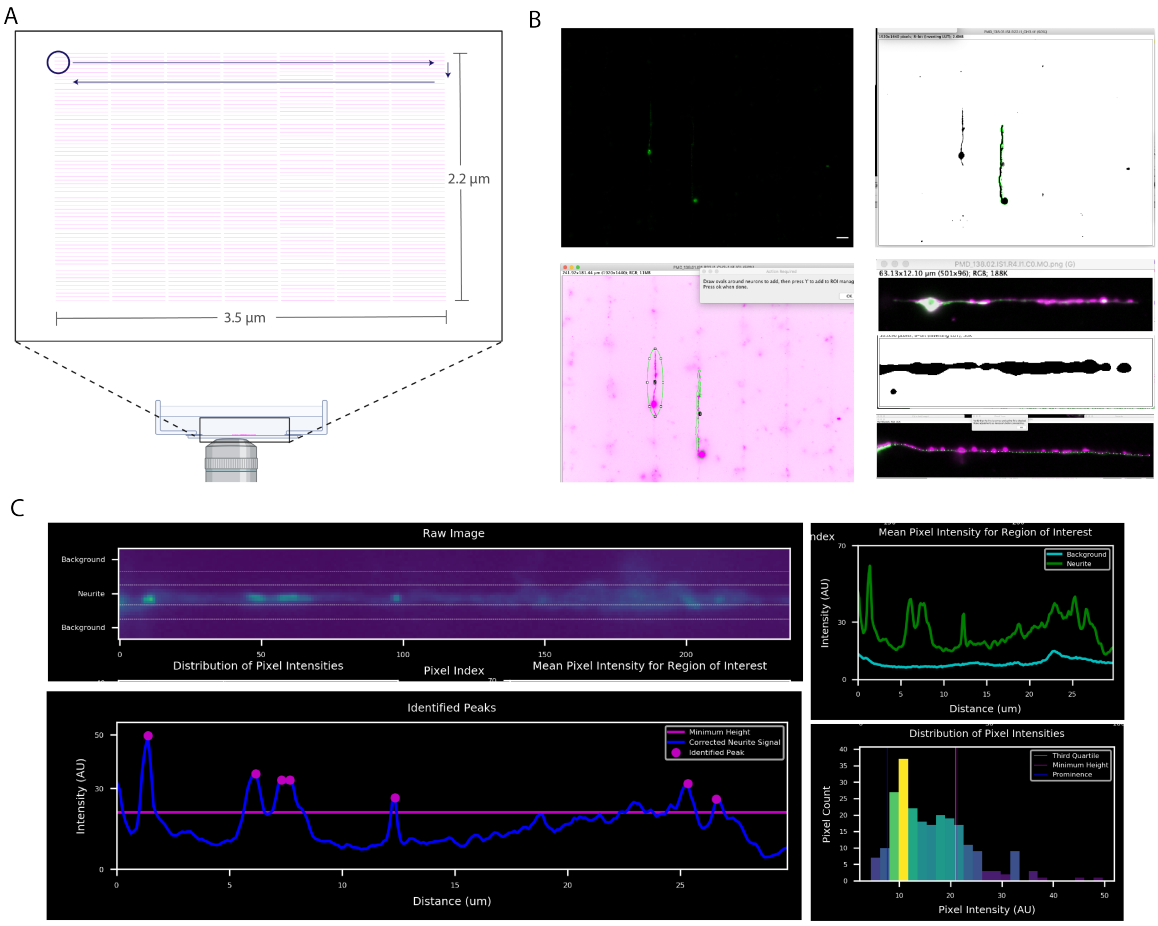


Figure B.1: **High-throughput quantification of mNG::MEC-4 distribution in isolated TRNs**. A) Systematic approach to scanning micropatterned areas, using predefined inclusion criteria for maximizing data acquisition while minimizing operator bias. B) Supervised workflow that full automates the mechanics of image pre-processing while providing access at each step for user intervention. Widefield images are processed for thresholding to generate cropped views and unique identifiers for each cell. Each cell is then divided into cell body and neurite. Straightened images of just the neurite are generated by line tracing. C) High-throughput classification of peaks in mNG::MEC-4 fluorescence intensity using custom, automated Peak Finder algorithm. Top left panel: Raw image of a TRN neurite. Top right panel: One-dimensional representations of raw image fluorescence intensities for background and neurite signals, respectively. Bottom left panel: Background subtracted 1D neurite signal with algorithm identified peaks annotated. Bottom right panel: Distribution of 1D pixel intensities for background subtracted neurite signal and annotations for algorithm calculated parameter values.

Figure B.2: **Representative montages of pre-processed images for quality control**. Montages of individual TRNs (A) and just the neurites (B) are generated by the supervised image pre-processing code. Users are given a choice to exclude individual cells from further processing during the generation of just neurite images, thus there is not always a one-to-one relation between the montages. These images are contrast and lookup-table optimized for visualization purposes and are not used for quantification. This represents a typical experiment throughput for a single cell culture prep that yields three technical replicates.

B.2 CropWCS.ijm

The first step in the image analysis process is to crop individual images of TRNs from the wide-field images that are acquired as described in Chapter 4 and shown in Figure B.2.A.

```
/*
 * PACC_CropWCS.ijm
 * JFRANCO
 * 20210301
 *
 * This macro is for cropping images of single cells to generate a
"Whole Cell Set" (WCS) from
 * larger images that are of entire fields of view at 60x. Due to the
small size of the cells,
 * the field of view is much larger than one cell and often contains
multiple cells and debris.
 * Cropping individual cells improves the results of the automated
neurite tracing step.
 *
 * Updates as of 20210301 are aimed at correcting for a chromatic
shift before cropping the images.
 * The correction will apply a transformation to the CH2 images (red)
so that they are aligned
 * to the CH3 images (green).
 */



/*
 ************************** MACRO PACC_CropWCS.ijm
*****************************
 */



/*
 *      ENTER USER SPECIFIC INFORMATION
 */
 // Requires user specific changes
dirCCPs = "/Users/nerdette/Google Drive/Research/WormSense/Data/
CCPs/";          // Path to main data subdirectory
channels = newArray("_CH3","_CH2", "_CH4");
                                                                    //
Channel names associated with raw tif files, order matters: GFP, RFP,
BF
// Close irrelevant images that might be open
run("Close All");


// Location for dialog boxes
x_d = 260;                                                          //
x position of dialog boxes
y_d = 125;                                                          //
y position of dialog boxes
```

```
// Location for images
x_iw = 260;                                                          //
x position of image window
y_iw = 300;                                                  // y
position of image window


/*
 * GET PREP, IMAGING, SAMPLING, AND WCS INFORMATION FROM USER
 */
//Check that results table is open
updateResults();

// Create dialog box
Dialog.create("Prep to analyze");
Dialog.addString("Prep Number",'138');
Dialog.addString("Whole Cell Set",'02');
Dialog.addString("Binning", '1x1');
Dialog.addNumber("Micron-to-Pixel Calibration", 0.126);
Dialog.setLocation(x_d,y_d);
Dialog.show();
// Read in values from dialog box
prepID = "CCP_"+Dialog.getString();
wcsID = "WCS_"+Dialog.getString();
binning = Dialog.getString();
cal = Dialog.getNumber();



/*
 * USE PROVIDED INFORMATION TO GENERATE PREP/WCS SPECIFIC PATHS AND
FILENAMES
 */

// Existing files
dirPrep = dirCCPs+prepID+"/";

        // CCP specific path
dirMetaD = dirPrep+"Metadata/";

        // Path to metadata files
dirRF= dirPrep+"Images/RawFrames/";
                                                                     //
Raw images to be cropped
dirMon = dirPrep+"Images/Montages/";
                                                                     //
Repository for storing generated montages
fnMDim = prepID+".MetaD.IM.csv";
// Files to be made during setup if this is the first run for WCS
dirWCS = dirPrep+"Images/Cropped/"+wcsID+"/";
```

```
                                                                    // Path to
WCS_## storage location for images that are cropped but retain raw
intensity values

dirROIs = dirPrep+"Locations/ROIs/"+wcsID+"/";
                                                                    // Path to
WCS_## storage location for ROIs identified during cropping
dirProc = dirPrep+"Images/Processed/"+wcsID+"/";
                                                        // Path to WCS_##
storage location for cropped and processed representation images
fnMDwcs = prepID+".MetaD."+wcsID+".csv";


/*
 * CHECK IF CROPPING FOR THIS WCS HAS BEEN STARTED.
 * & SETUP METADATA
 */
setupDir(dirWCS, dirProc, dirROIs, dirRF, dirMetaD, fnMDwcs);


/*
 * BEGIN CROPPING PROCESS
 */
choice = getUserChoice();
index = 0;
while (choice != 'EXIT') {
        // Start the analysis clock
        getDateAndTime(year, month, dayOfWeek, dayOfMonth, hourSt,
minuteSt, secondSt, msecSt);
        strDateTime = getTimeStamp();

        // Get index of what image to open next
        index = findIndex(index);
                                                                    // Fx uses
index to iterate through ResultsTable and returns first instance where
wcs_processed=NO
        imname = getResultString("image_name", index);

        // Set date and time of analysis
        timestamp = getTimeStamp();
        setResult("timestamp", index, timestamp);

        // Open unprocessed image & adjust for display
        fnIM_GFP = imname+channels[0]+".tif";
        pathToIm = dirRF+imname+"/";
        open(pathToIm+fnIM_GFP);
        setLocation(x_iw, y_iw);
        run("Invert");
        run("Enhance Contrast...", "saturated=0.1");
```

```
        // Verify if cropping should proceed
        options = Array.concat("YES", "NO");
        Dialog.create("PROCEED WITH CROPPING CHECK");
        Dialog.addChoice("Continue to image cropping?", options);
        Dialog.setLocation(x_d,y_d);
        Dialog.show();
        check = Dialog.getChoice();

        // Case where user wants to proceed with cropping
        if (check == 'YES'){
                // Threshold image
                fixThresh = procToBinary(fnIM_GFP);
                setResult("fix_threshold", index, fixThresh);
                updateResults();
                // Get ROIs of neurons
                fixROIs = getROIs(fnIM_GFP, pathToIm, dirROIs);
                roicount = roiManager("count");
                // Taking a maximum of 5 neurons per image
                if (roicount>5) {
                        roicount=5;
                }
                setResult("add_rois", index, fixROIs);
                updateResults();

                /*
                 * ITERATE THROUGH ROIs TO CROP IMAGES
                 */
                for (i = 0; i < roicount; i++) {
                        /*
                         * Make a processed jpg of mNG image for
review & display purposes
                         */
                        open(pathToIm+fnIM_GFP);
                        setLocation(x_iw, y_iw);
                        run("Set Scale...", "distance=1 known="+cal+"
pixel=1 unit=um global");
                        setMinAndMax(0, 50);
                        run("Invert");
                        roiManager("Select", i);
                        run("To Bounding Box");
                        run("Enlarge...", "enlarge=25 pixel");
                        run("Crop");
                        fnProc = imname+".C"+toString(i)+".pp.jpg";
                        // Rotate image to get cell body on left side
                        autoRot = rotateImage(fnIM_GFP);
                        // Check with user if rotation needs to be
fixed
                        userRot = fixRotate(fnIM_GFP);
                        // Update Results Table to track work
                        setResult("ar_c"+toString(i),index,autoRot);
```

```
                        setResult("ur_c"+toString(i),index,userRot);
                        updateResults();
                        // Add scale bar and save image
                        run("Scale Bar...", "width=5 height=4 font=20
color=White background=None location=[Lower Right] hide");
                        saveAs("Jpeg",dirProc+fnProc);
                        close();

                        /*
                         * Make a set of cropped images for every
channel with raw intensity values
                         */
                        // Open raw .tif files for every channel
                        for( j = 0; j < channels.length ; j++) {
                                file =
dirRF+imname+"/"+imname+channels[j]+".tif";
                                open(file);
                                run("Set Scale...", "distance=0.00
known=0.00 pixel=1 unit=pixel");
                                // If this is the ch2 image it will
need to be translated prior to stack formation
                                if (j==1) {
                                        run("TransformJ Translate",
"x-distance=1.5 y-distance=.5 z-distance=0.0 interpolation=Linear
background=0.0");

selectWindow(imname+channels[j]+".tif");
                                        close();

selectWindow(imname+channels[j]+".tif translated");
                                        rename(imname+channels[j]
+".tif");
                                }
                                setLocation(x_iw, y_iw);
                        }
                        //  Create image stack
                        run("Images to Stack", "name=" + imname + "
title=[] use");
                        selectImage(imname);
                        roiManager("Select", i);
                        run("To Bounding Box");
                        run("Enlarge...", "enlarge=25 pixel");
                        run("Crop");
                        // Rotate image to get cell body on left side
                        run("Rotate 90 Degrees Right");
                        if ((autoRot=="SKIP" && userRot=="FIX") ||
(autoRot=="ROT" && userRot=="SKIP")) {
                                run("Rotate 90 Degrees Right");
                                run("Rotate 90 Degrees Right");
                        }
```

```
                        // Save all images
                        savename = imname+".C"+toString(i)+".WC";
                        saveinfo = "["+dirWCS+"]";
                        run("Image Sequence... ", "format=TIFF
name="+savename+" digits=1 save="+saveinfo);
                        selectImage(imname);
                        close();
                        }

                }else {
                        // Case where user did not want to crop the
image
                        selectWindow(fnIM_GFP);
                        close();
                        setResult("fix_threshold", index, "NA");
                        setResult("add_rois", index, "NA");
                        updateResults();
                }

                // Stop the analysis clock and calculate duration
                getDateAndTime(year, month, dayOfWeek, dayOfMonth,
hourFi, minuteFi, secondFi, msecFi);
                timeSt = (hourSt*3600) + (minuteSt*60) + secondSt +
(msecSt/1000);
                timeFi = (hourFi*3600) + (minuteFi*60) + secondFi +
(msecFi/1000);
                duration = timeFi-timeSt;
                setResult("duration", index, duration);
                updateResults();

                // Update wcs processed and update metadata file
                setResult("wcs_processed", index, 'YES');
                updateResults();
                selectWindow("Results");
                saveAs("Results", dirMetaD+fnMDwcs);

                /*
                 * Update index & user choice to continue cropping
                 */
                index++;
                if (index < nResults) {
                        choice = getUserChoice();
                }else{
                        waitForUser("All images have been cropped.
\n"+
                                                "The macro will
after montage generation.");
                        choice = 'EXIT';
                }
```

```
}

/*
 * Make montage of cropped images in WCS if user so chooses

 */
options = Array.concat("NO", "YES");
Dialog.create("MAKE MONTAGE BEFORE EXITING?");
Dialog.addChoice("Proceed to montage generation?", options);
Dialog.addMessage("Montage will be made for all pp .jpgs\nin Processed
folder");
Dialog.setLocation(x_d,y_d);
Dialog.show();
choiceMM = Dialog.getChoice();
if (choiceMM == "YES") {
        makeMontage(dirMon, dirProc, prepID, wcsID);
}

/*
 * ****************************  MACRO END
*****************************
 */



/*
 * ************************               FUNCTION DEFINITIONS
        **********************
 */
function getTimeStamp(){
//      FUNCTION GETS TIME STAMP FOR METADATA
        print("\\Clear");
        getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour,
minute, second, msec);
        month++;
        if (month<10) {month = "0"+toString(month);}
        if (dayOfMonth<10) {month = "0"+toString(dayOfMonth);}
        date = toString(year)+ month + dayOfMonth;
        if (hour<10) {hour = "0"+toString(hour);}
        if (minute<10) {minute = "0"+toString(minute);}
        time = toString(hour)+"h"+toString(minute) +"m";
        arrDateTime = Array.concat(date + "_"+ time);
        Array.print(arrDateTime);
        strDateTime = toString(getInfo("log"));
        strDateTime = substring(strDateTime, 0,
lengthOf(strDateTime)-1);
        return strDateTime;
}
```

```
function setupDir(dirWCS, dirProc, dirROIs, dirRF, dirMetaD, fnMDwcs){
// CHECKS IF CROPPING FOR THIS WCS HAS ALREADY BEEN STARTED
//     IF NO => Setup dirWCS, dirProc, dirROIs, and fnMDwcs
//         IF YES => Skip directory setup
         if (File.exists(dirWCS)){
                 // If the set exists, verify continuation
                 Dialog.create("Error");
                 Dialog.addCheckbox("WCS exists, continue? ",true);
                 Dialog.setLocation(x_d,y_d);
                 Dialog.show();
                 inBoolean = Dialog.getCheckbox();
                 if (!inBoolean){
                         //If the user doesn't want to continue, get
out
                         print("Note      Macro ended by user
choice");
                         exit("Note       Please restart & specify the
correct WCS ID");
                 }else {
                         // If user wants to continue, load metadata
                         setupMD(dirMetaD+fnMDwcs);
                 }
         }else{
                 // If WCS doesn't exist, initilize relevant items
                 File.makeDirectory(dirWCS);
                 File.makeDirectory(dirProc);
                 File.makeDirectory(dirROIs);

                 // Run FX for initializing results table for storing
& generating metadata .csv file
                 initResTable(dirRF, dirMetaD+fnMDwcs);
         }
}

function initResTable(dirRF, fPathMD){
// FUNCTION RUNS IF THIS IS THE FIRST TIME SETTING UP WCS.
//    Fx setsup Results table with relevant column names

         // Setup MD Results Table based on list of images in RawFrames
directory
         run("Clear Results");
         imlist = getFileList(dirRF);
         for (i = 0; i < lengthOf(imlist); i++) {
                 fn = substring(imlist[i], 0,lastIndexOf(imlist[i],
'/'));
                 setResult("image_name", i, fn);
                 setResult("timestamp", i, 'TBD');
                 setResult("wcs_processed", i, 'NO');
                 setResult("duration", i, 'TBD');
                 setResult("fix_threshold", i, 'TBD');
```

```
                setResult("add_rois",i,'TBD');
                for (j=0; j < 5; j++) {
                        setResult("ar_c"+toString(j),i,'NA');
                        setResult("ur_c"+toString(j),i,'NA');
                }
                updateResults();
        }

        // Immediately create a saved copy of the MD file
        selectWindow("Results");
        saveAs("Results", fPathMD);
}

function setupMD(fPath){
// FX Reads in csv file and setsup info as a Results table
//    In particular, reads in CCP_###.MetaD.WCS_##.csv
        run("Clear Results");
        lineseparator = "\n";
        cellseparator = ",\t";

        // copies the whole RT to an array of lines
        lines=split(File.openAsString(fPath), lineseparator);

        // recreates the columns headers
        labels=split(lines[0], cellseparator);
        if (labels[0]==" "){
                k=1; // it is an ImageJ Results table, skip first
column
        }else{
        k=0; // it is not a Results table, load all columns
        }
        for (j=k; j<labels.length; j++)
                setResult(labels[j],0,0);
                // dispatches the data into the new RT
        run("Clear Results");
        for (i=1; i<lines.length; i++) {
                items=split(lines[i], cellseparator);
        for (j=k; j<items.length; j++)
                setResult(labels[j],i-1,items[j]);
        }
        updateResults();
}

function getUserChoice() {
// SETUP GUI FOR USER TO EITHER QUIT OR SELECT NEXT IMAGE FOLDER
        options = Array.concat("CONTINUE", "EXIT");
        Dialog.create("Puncta Analysis PreProcessing");
        Dialog.addChoice("To exit the cropping macro, select 'Exit",
options);
        Dialog.setLocation(x_d,y_d);
```

```
        Dialog.show();

        // GET AND RETURN USER INPUT ONCE USER HITS 'OK' ON DIALOG BOX
        return Dialog.getChoice();
}

function findIndex(index) {
        proc = getResultString("wcs_processed", index);
        while(proc == 'YES'){
                index++;
                if(index < nResults){
                proc = getResultString("wcs_processed", index);
                }
                if(index >= nResults){
                        exit("All images have been cropped. Macro
will exit.\n"+
                        "Restart and skip cropping to make montage");

                }
        }
        return index;
}

function procToBinary(imname) {
        selectWindow(imname);
        run("Subtract Background...", "rolling=30 light separate
sliding disable");
        run("Duplicate...", "title=Duplicate");
        selectWindow(imname);
        setOption("BlackBackground", false);
        run("Convert to Mask");
        run("Erode");
        run("Dilate");

        // Verify with user that thresholding is good and doesn't need
adjustment
        choice = fixThreshold("Does thresholding need to be
adjusted?");
        selectImage("Duplicate");
        close();
        return choice;
}

function fixThreshold(message){
// CHECK IF CURRENT THRESHOLDING IS SUFFICIENT OR IF USER NEEDS TO FIX
IT
        optFix = Array.concat("SKIP","FIX");
        Dialog.create("FIX THRESHOLDING");
        Dialog.addMessage(message);
        Dialog.addChoice("Do you want to manually fix the IDd blobs?",
```

```
optFix);
        Dialog.setLocation(x_d,y_d);
        Dialog.show();
        choiceFix = Dialog.getChoice();
        if (choiceFix == "FIX") {
                setForegroundColor(255, 252, 255);
                setTool("Paintbrush Tool");
                waitForUser("Use white paintbrush to make cuts\n"+
                        "then press enter.");
                setTool("Paintbrush Tool");
                setForegroundColor(0, 0, 0);
                waitForUser("Use black paintbrush to fill in gaps\n"+
                        "then press enter.");
        }
        return choiceFix;
}

function getROIs(imname, pathToIm, dirROIs) {
// ANALYZE PARTICLES TO REDUCE ROIs TO ONLY        THOSE THAT ARE
NEURONS
        roiManager("reset");
        selectImage(imname);
        run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
        run("Analyze Particles...", "size=5000-Infinity
circularity=0.00-0.20 solidity=0.00"+
        "display exclude add");
        choiceAdd = addROIs(imname, pathToIm);
        if (roiManager("count")>0){
                roiManager("save", dirROIs+"/"+imname+".zip");
        }
        selectImage(imname);
        close();
        return choiceAdd;
}


function addROIs(imname , pathToIm ){
// CHECK IF THERE ARE ADDITIONAL ROIs that need to be added
        optAdd = Array.concat("SKIP","ADD");
        Dialog.create("ADD ROIs");
        Dialog.addMessage("Are there additional ROIs to identify?");
        Dialog.addChoice("Skip step or add new ROIs?", optAdd);
        Dialog.setLocation(x_d,y_d);
        Dialog.show();
        choiceAdd = Dialog.getChoice();
        if (choiceAdd == "ADD") {
                open(pathToIm+imname );
                setLocation(x_iw, y_iw);
                run("Set Scale...", "distance=1 known="+cal+" pixel=1
unit=um global");
```

```
                run("Invert");
                run("Enhance Contrast...", "saturated=0.1");
                setTool("oval");
                waitForUser("Draw ovals around neurons to add, then
press 't' to add to ROI manager.\n"+
                                                "Press ok when done.");
                close();
        }
        return choiceAdd;
}

function rotateImage(imhandle){
// FUNCTION ATTEMPTS TO AUTOMATICALLY ROTATE THE IMAGE SO THAT THE
CELL BODY
// IS ON THE LEFT SIDE OF THE IMAGE

        /*
         * Attempt #1 to automate the image rotation process
         */
        selectImage(imhandle);
        run("Rotate 90 Degrees Right");
        getDimensions(width, height, channels, slices, frames);
        widthNew = width/2;
        makeRectangle(0, 0, widthNew, height);
        lsFeretX = getValue("FeretX");
        makeRectangle(widthNew, 0, width, height);
        rsFeretX = getValue("FeretX");
        if (lsFeretX > rsFeretX){
                selectImage(imhandle);
                run("Rotate 90 Degrees Right");
                run("Rotate 90 Degrees Right");
                return "ROT";
        }else{
                return "SKIP";
        }
}

function fixRotate(imhandle){
// FUNCTION CHECKS WITH USER IF IMAGE ROTATION NEEDS TO BE FIXED AND
RETURNS RESPONSE
        Dialog.create("ROTATION CHECK");
        Dialog.addMessage("The cell body should be on the left side
with the\n"+
                "neurite extending to the right.");
        Dialog.addCheckbox("Repeat transformation?", false);
        Dialog.setLocation(x_d,y_d);
        Dialog.show();
        choiceRot = Dialog.getCheckbox();
        if (choiceRot) {
                // If rotation needs to be fixed, then it means image
```

```
does not need
                // to be rotated beyond the original transformation
                run("Rotate 90 Degrees Right");
                run("Rotate 90 Degrees Right");
                return "FIX";
        }else{
                // If rotation doesn't need to be fixed, then the
image did need to be
                // rotated beyond the original transformation
                return "SKIP";
        }
}


function makeMontage(dirMon, dirProc, prepID, wcsID){
        fnMontage = prepID+".Montage.AllCropped."+wcsID+".jpg";
        imlist = getFileList(dirProc);
        for (i = 0; i < lengthOf(imlist); i++) {
                open(dirProc+imlist[i]);
                setLocation(x_iw, y_iw);
        }
        rows = round(lengthOf(imlist)/2);
        run("Images to Stack", "method=[Copy (center)] fnMontage
title=[] use");
        run("Make Montage...", "columns=2 rows="+toString(rows)+"
scale=0.50 label");
        saveAs("Jpeg",dirMon+fnMontage);
}
```

B.3 NeuTrace.ijm

The second step in the image analysis process is to genearte a set of straightened neurite views from the images cropped in "Crop Whole Cell Set." These "just neurite" images (Figure B.2.B) are read in automatically by the Peak Finder code for TRN morphological analysis and quantification of MEC-4 subcellular localization.

```
/*
 * IJM_NeuTrace.ijm
 * JFRANCO
 * 20210226
 *
 * The purpose of this macro is to take in an image, process it for
thresholding, save the ID'd ROI,
 * and then use it to sample other images.
 *
 * A key difference between this macro and the original one made for
PACC is that it uses a merged overlay to determine
 * the region to trace rather than the single channel image.
 */

/*
 ************************** MACRO PACC_NeuTrace.ijm
*****************************
 */
run("Close All");

/*
 *      ENTER USER SPECIFIC INFORMATION
 */
 // Requires user specific changes
dirCCPs = "/Users/nerdette/Google Drive/Research/WormSense/Data/
CCPs/";
channels = newArray("_CH3","_CH2", "_CH4");
                                                                    //
Channel names associated with raw tif files, order matters: GFP, RFP,
BF
// Close irrelevant images that might be open
run("Close All");


// Location for dialog boxes
x_d = 260;                                                          //
x position of dialog boxes
y_d = 125;                                                          //
y position of dialog boxes
// Location for images
x_iw = 260;                                                         //
x position of image window
y_iw = 300;                                                 // y
position of image window
// Location setup for ROI manager
x_roi = 0;
y_roi = 100;
run("ROI Manager...");
selectWindow("ROI Manager");
setLocation(x_roi, y_roi);
```

```
/*
 * GET PREP, IMAGING, SAMPLING, AND WCS INFORMATION FROM USER
 */
// Create dialog box
Dialog.create("Prep to analyze");
Dialog.addString("Prep Number",'138');
Dialog.addString("Whole Cell Set",'02');
Dialog.addString("Neurite Set",'01');
Dialog.addString("Binning", '1x1');
Dialog.addNumber("Micron-to-Pixel Calibration", 0.126);
Dialog.addNumber("Physical Sampling Space (um)", 4);
Dialog.setLocation(x_d,y_d);
Dialog.show();
// Read in values from dialog box
prepID = "CCP_"+Dialog.getString();
wcs = Dialog.getString();
wcsID = "WCS_"+wcs;
nsID = "NS_"+wcs+"."+Dialog.getString();
binning = Dialog.getString();
cal = Dialog.getNumber();
samplespace = Dialog.getNumber();
                                                                        //
Physical sampling space
linewidth = round(samplespace/cal);

        // How thick of a line (in pixels) to draw around main trace
line

/*
 * USE PROVIDED INFORMATION TO GENERATE PREP/WCS SPECIFIC PATHS AND
FILENAMES
 */
 // Existing files
dirPrep = dirCCPs+prepID+"/";

        // CCP specific path
dirMetaD = dirPrep+"Metadata/";

        // Path to metadata files
dirWCS = dirPrep+"Images/Cropped/"+wcsID+"/";
                                                                // Path to
WCS files to use for tracing
dirProcWCS = dirPrep+"Images/Processed/"+wcsID+".MO/";
                                                                // Path to
processed WCS files to use for display
dirROIs = dirPrep+"Locations/ROIs/"+nsID+"/";
                                                                // Path to
WCS_## storage location for ROIs identified during cropping
dirMon = dirPrep+"Images/Montages/";
```

```
                                                                        //
Repository for storing generated montages
// Files to be made during setup if this is the first run for WCS
dirNS = dirPrep+"Images/Cropped/"+nsID+"/";

                                    // Path to NS_## storage location
for images that are traced but retain raw intensity values


dirProcJN = dirPrep+"Images/Processed/"+nsID+"/";
                                                      // Path to NS_##
storage location for traced and processed representation images
fnMDns = prepID+".MetaD."+nsID+".csv";
                                                                        //
File: Metadata for NS

/*
 * CHECK IF CROPPING FOR THIS WCS HAS BEEN STARTED.
 * & SETUP METADATA
 */
setupDir(dirProcWCS, dirROIs, dirNS, dirProcJN, dirMetaD, fnMDns);

/*
 * BEGIN TRACING PROCESS
 */
choice = getUserChoice();
index = 0;
while (choice != 'EXIT') {
        // Start the analysis clock
        getDateAndTime(year, month, dayOfWeek, dayOfMonth, hourSt,
minuteSt, secondSt, msecSt);
        strDateTime = getTimeStamp();
        // Get index of what image to open next
        index = findIndex(index);
                                                              // Fx uses
index to iterate through ResultsTable and returns first instance where
ns_processed=NO
        imname = getResultString("image_name", index);

        /*
        ********   OPEN PROCESSED DISPLAY IMAGE TO TEST FOR INCLUSION
***********
        * Images are processed in the order they appear in imlist,
therefore, indices should match up with what's in the
        *       metadata and neurite tracing can be resumed using
this approach
        */
        imnamePP = imname+".MO.png";
        open(dirProcWCS+imnamePP);
        setLocation(x_iw, y_iw);
```

```
	/*
	******** INCLUSION CHECK   ***********
	*/
	Dialog.create("Exlude TRN From Analysis?");
	Dialog.setLocation(x_d,y_d);
	Dialog.addCheckbox("Exclude cell fron neurite set?", false);
	Dialog.addChoice("Exclusion Criteria:", newArray("Bipolar",
"Psuedo-Bipolar", "No Neurites", "Other"));
	Dialog.show();
	exclude = Dialog.getCheckbox();
	reason = Dialog.getChoice();
	if(exclude){
		// CASE WHERE USER WANTS TO EXCLUDE CELL FROM NS
		inorout = "Exclude";
		selectWindow(imnamePP);
		fixes = newArray(4);
		fixes[0]=0;
		close();
	}else{
		// CASE WHERE USER WANTS TO INCLUDE CELL IN NS
		inorout = "Include";
		reason = "0";
		fixes = neuTrace(channels, imname, dirProcWCS,
dirROIs, dirWCS, linewidth, cal, dirProcJN, dirNS, imnamePP);
	}

	/*
	 * STOP ANALYSIS CLOCK & CALCULATE PROCESSING TIME
	 */
	getDateAndTime(year, month, dayOfWeek, dayOfMonth, hourFi,
minuteFi, secondFi, msecFi);
	timeSt = (hourSt*3600) + (minuteSt*60) + secondSt + (msecSt/
1000);
	timeFi = (hourFi*3600) + (minuteFi*60) + secondFi + (msecFi/
1000);
	duration = timeFi-timeSt;

	setResult("timestamp", index, strDateTime);
	setResult("ns_processed", index, 'YES');
	setResult("in_or_out",index,inorout);
	setResult("exclusion_reason",index,reason);
	setResult("duration", index, duration);
	setResult("cb_rois",index,fixes[0]);
	setResult("neu_rois",index,fixes[1]);
	cbFix = fixes[2];
	setResult("fix_cb",index,cbFix);
	neuFix = fixes[3];
	setResult("fix_neu",index,neuFix);
	updateResults();
```

```
        // Save results on each run in case macro crashes
        selectWindow("Results");
        saveAs("Results", dirMetaD+fnMDns);

        /*
         * Update index & user choice to continue cropping
         */
        index++;
        if (index < nResults) {
                choice = getUserChoice();
        }else{
                waitForUser("All images have been traced.\n"+
                                        "The macro will after
montage generation.");
                choice = 'EXIT';
        }
}

/*
********   MONTAGE GENERATION (IF APPLICABLE) ***********
*/
options = Array.concat("NO", "YES");
Dialog.create("MAKE MONTAGE BEFORE EXITING?");
Dialog.setLocation(x_d,y_d);
Dialog.addChoice("Proceed to montage generation?", options);
Dialog.addMessage("Montage will be made for all pp .jpgs\nin Processed
folder");
Dialog.show();
choiceMM = Dialog.getChoice();

if (choiceMM == "YES") {
        makeMontage(dirMon, dirProcJN, prepID, nsID);
}

exit("bye bye");
/*
 * ******************************  MACRO END
*******************************
 */



/*
 * ************************                FUNCTION DEFINITIONS
        ***********************
 */
function getTimeStamp(){
        print("\\Clear");
        getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour,
```

```
minute, second, msec);
        month++;
        if (month<10) {month = "0"+toString(month);}
        if (dayOfMonth<10) {month = "0"+toString(dayOfMonth);}
        date = toString(year)+ month + dayOfMonth;
        if (hour<10) {hour = "0"+toString(hour);}
        if (minute<10) {minute = "0"+toString(minute);}
        time = toString(hour)+"h"+toString(minute) +"m";
        arrDateTime = Array.concat(date + "_"+ time);
        Array.print(arrDateTime);
        strDateTime = toString(getInfo("log"));
        strDateTime = substring(strDateTime, 0,
lengthOf(strDateTime)-1);
        return strDateTime;
}

// Make sure dirWCS exists and if not throw error, make dirNS, make
dirProcJN, initialize results table for metaD and save; if ns has been
started load metad
function setupDir(dirProcWCS, dirROIs, dirNS, dirProcJN, dirMetaD,
fnMDns){
// VERIFIES THAT SPECIFIED WCS EXISTS; IF NOT => THROW ERROR
// CHECKS IF NEURITE TRACING HAS ALREADY BEEN STARTED
//              IF NO => SETUP dirNS, dirProcJN, and fnMDns
//              IF YES => LOAD fnMDns
        if(File.exists(dirWCS)==0){
                exit("Error; WCS does not exist. Please restart the
macro and indicate a valid WCS.");
        }
        if (File.exists(dirNS)){
                // If the neurite set has already been started,
verify continuation
                Dialog.create("NS EXISTS CHECK");
                Dialog.setLocation(x_d,y_d);
                Dialog.addCheckbox("NS exists, resume processing?
",true);
                Dialog.show();
                inBoolean = Dialog.getCheckbox();
                if (!inBoolean){
                        // User does not want to resume processing,
exit macro
                        exit("Exit by user choice; Incorrect NS_ID");


                }else {
                        // User wants to resume processing, load
metadata
                        setupMD(dirMetaD+fnMDns);
```

```
                }
        }else{
                // If dirNS doesn't exist, initialize relevant items
                File.makeDirectory(dirROIs);
                File.makeDirectory(dirNS);
                File.makeDirectory(dirProcJN);
                // Run FX for initializing results table for storing
& generating metadata .csv file
                initResTable(dirProcWCS, dirMetaD+fnMDns);
        }
}

function initResTable(dirProcWCS, fPathMD){
// FUNCTION RUNS IF THIS IS THE FIRST TIME SETTING UP NS.
//    Fx setsup Results table with relevant column names

        // Setup MD Results Table based on list of images in
ProcessedWCS directory
        run("Clear Results");
        imlist = getFileList(dirProcWCS);

        for (i = 0; i < lengthOf(imlist); i++) {
                fn = substring(imlist[i], 0, lastIndexOf(imlist[i],
".MO"));
                setResult("image_name", i, fn);
                setResult("timestamp", i, 'TBD');
                setResult("ns_processed", i, 'NO');
                setResult("line_width", i, linewidth);
                setResult("in_or_out",i,'TBD');
                setResult("exclusion_reason",i,'TBD');
                setResult("duration", i, 'TBD');
                setResult("cb_rois",i,'TBD');
                setResult("neu_rois",i,'TBD');
                setResult("fix_cb",i,'TBD');
                setResult("fix_neu",i,'TBD');
                updateResults();
        }
        // Immediately create a saved copy of the MD file
        selectWindow("Results");
        saveAs("Results", fPathMD);
}

function setupMD(fPath){
// FX Reads in csv file and setsup info as a Results table
//    In particular, reads in CCP_###.MetaD.NS_##.##.csv
        lineseparator = "\n";
        cellseparator = ",\t";

        // copies the whole RT to an array of lines
        lines=split(File.openAsString(fPath), lineseparator);
```

```
        // recreates the columns headers
        labels=split(lines[0], cellseparator);
        if (labels[0]==" "){
                k=1; // it is an ImageJ Results table, skip first
column
        }else{
        k=0; // it is not a Results table, load all columns
        }
        for (j=k; j<labels.length; j++)
                setResult(labels[j],0,0);
                // dispatches the data into the new RT
        run("Clear Results");
        for (i=1; i<lines.length; i++) {
                items=split(lines[i], cellseparator);
        for (j=k; j<items.length; j++)
                setResult(labels[j],i-1,items[j]);
        }
        updateResults();
}

function getUserChoice() {
// SETUP GUI FOR USER TO EITHER QUIT OR SELECT NEXT IMAGE FOLDER
        options = Array.concat("CONTINUE", "EXIT");
        Dialog.create("NEURITE TRACING PROCESS");
        Dialog.setLocation(x_d,y_d);
        Dialog.addChoice("To exit the neurite tracing macro, select
'Exit", options);
        Dialog.show();

        // GET AND RETURN USER INPUT ONCE USER HITS 'OK' ON DIALOG BOX
        return Dialog.getChoice();
}

function findIndex(index) {
        proc = getResultString("ns_processed", index);
        while(proc == 'YES'){
                index++;
                if(index < nResults){
                proc = getResultString("ns_processed", index);
                }
                if(index >= nResults){
                        exit("All images have been traced. Macro will
exit.\n"+
                        "Restart and skip tracing to make montage");

                }
        }
        return index;
}
```

```
function neuTrace(channels, imnameBASE, dirProcWCS, dirROIs, dirWCS,
linewidth, cal, dirProcJN, dirNS, imnamePP){
// FUNCTION TRACES NEURITE AND SAVES IMAGES

        roiManager("reset");
        fixInfo = newArray(4);

        // Open image & setup filenames for future purposes
        imnamesWC = newArray(3);
        imnamesJN = newArray(3);
        for (i = 0; i < lengthOf(channels); i++) {
                imnamesWC[i] = imnameBASE+".WC"+toString(i)+".tif";
                imnamesJN[i] = imnameBASE+".JN"+toString(i)+".tif";
        }

        selectWindow(imnamePP);
        run("Set Scale...", "distance=0 known=0 pixel=1 unit=pixel");
        run("Invert");
        run("Enhance Contrast...", "saturated=0.1");
        run("Make Binary");
        run("Erode");
        run("Erode");
        run("Erode");

        // Check if user needs to fix anything
        message = "Fix threshold for cell body ID";
        fixCB = fixCheck(message);
        fixInfo[2] = fixCB;

        /*
         *  ANALYZE  PARTICLES TO ID CELL BODY BLOB
         *  Also verifies that correct ROI was identified and stores
information about accuracy.
         */
        run("Analyze Particles...", "size=250-Infinity
circularity=0.3-1.00 solidity=0.00"+
                 "exclude include add");

        fixInfo[0] = roiManager("count");

                        // Count the number of blobs detected for
cell body
        roiCB = fixInfo[0];
        if(roiCB < 1){
                waitForUser("CB not detected. Please draw and add to
the ROI manager");
        }
        if(roiCB >1){
                waitForUser("Multiple CBs detected. Please delete
```

```
wrong ROIs from the manager");
        }
        if(roiCB == 1){
                optCB = Array.concat("YES","NO");
                Dialog.create("Verify CB");
                Dialog.addMessage("CB detected. Please verify its
accuracy before proceeding");
                Dialog.addChoice("Is CB detected correct?", optCB);
                Dialog.setLocation(x_d,y_d);
                Dialog.show();
                choiceCB = Dialog.getChoice();
                if (choiceCB == 'NO') {
                        roiCB=0;
                        waitForUser("Please manually draw the correct
ROI and add to ROI manager.");
                }
        }
        // The following steps are to select the region that is the
cell body and then create
        // an ROI that does not include the cell body so that the
original image can be cropped
        // to only the neurite region.
        selectWindow(imnamePP);
        setLocation(x_iw, y_iw);
        roiManager("Select", 0);
                // 1) Select the ROI containing the cell body
        getSelectionBounds(coord_x, coord_y, w_s, h_s);    // 2) Find
a bounding box for the ROI
        selectImage(imnamePP);

        getDimensions(w_i, h_i, chan, slices, frames);             //
3) Generate the ROI for neurite based on cell body ROI
        x_new = coord_x+w_s;
        w_new = w_i-x_new-1;
        makeRectangle(x_new, 0, w_new, h_i);
        roiManager("Add");
                                // 4) Add the new ROI to the manager
        close();
                                // 5) Housekeeping: close the
processed image
        open(dirProcWCS+imnamePP);
                        // 6) Open new instance of merge image for
neurite ID
        roiManager("Select", 1);
        run("Crop");
        roiManager("reset");


        // ****  neurite id  ****
        // Preprocess for thresholding
```

```
        run("Find Edges");
        run("Smooth");
        run("Maximum...", "radius=2");
        run("Gaussian Blur...");
        run("8-bit");

        // Threshold and filter out irrelevant particles
        run("Auto Threshold", "method=MaxEntropy white");
        run("Dilate");
        run("Convert to Mask");

        // Here is a good place to see if the mask needs to be
adjusted to improve the skeleton
        selectWindow(imnamePP);
        setLocation(x_iw, y_iw);
        run("In [+]");
        run("In [+]");
        run("Scale to Fit");


        // Check if user needs to fix anything - this is where it
would be good to have a
        //    reference image available to the user to help them
verify threshold
        message = "Fix threshold for neurite ID";
        open(dirProcWCS+imnamePP);
                        //Open new instance of merge image to help
with fixing the threshold
        selectWindow(imnamePP);
        fixNeu = fixCheck(message);
        fixInfo[3] = fixNeu;

        run("Analyze Particles...", "size=1000-15000
circularity=0.00-0.5 solidity=0.00 AR=5-Infinity"+
                 "exclude include add");

        // Verify that analyze particles is only returning one region
        fixInfo[1] = roiManager("count");
                                                                    //
Count the number of blobs detected for neurite
        roiNeu = fixInfo[1];


        if(roiNeu < 1){
                waitForUser("Neurite not detected. Please draw and
add to the ROI manager");
        }
        if(roiNeu >1){
                waitForUser("Multiple CBs detected. Please delete
wrong ROIs from the manager");
```

```
        }
        if(roiNeu == 1){
                optNeu = Array.concat("YES","NO");
                Dialog.create("Verify Neurite ROI");
                Dialog.addMessage("Neurite detected. Please verify
its accuracy before proceeding");
                Dialog.addChoice("Is the detected neurite blob
correct?", optNeu);
                Dialog.setLocation(x_d,y_d);
                Dialog.show();
                choiceNeu = Dialog.getChoice();
                if (choiceNeu == 'NO') {
                        roiNeu=1;
                        waitForUser("Please manually draw the correct
ROI and add to ROI manager.");
                }
        }

        roiManager("Select", 0);
        run("Clear Outside");
        run("Invert");
        run("Convert to Mask");

        // Smooth out the ID'd line
        run("Skeletonize");
        run("Smooth");
        run("Smooth");
        roiManager("Delete");
        run("Convert to Mask");
        run("Create Selection");
        run("Fit Spline");
        run("Clear Outside");
        run("Invert");
        run("Convert to Mask");
        run("Skeletonize");

        // Get LGCs from the mask
        run("Profile Plot Options...",
                                                                //Important
to set options to get values from line graph
             "list interpolate draw");
        run("Analyze Line Graph");
        Plot.getValues(lgcX, lgcY);

        // Need to reduce the number of x and y coordinates to reduce
the noise.
        lenXOld = lengthOf(lgcX);
        lenXNew = round(lenXOld/3)+2;
        lineX = newArray(lenXNew);
        lineY = newArray(lenXNew);
```

```
        for (i = 0; i < 3; i++) {
                lineX[i] = i*3;
                lineY[i] = lgcY[4];
        }
        j=3;
        // i starts at sixth index (~10th pixel) because that's where
first for-loop leaves off
        // only every third XY coordinate from original line is used
        // it's when j = lenXNew-1 that we know we're about to set the
very last set of coordinates for the new array
        for (i=6; j<=lenXNew-1; i=i+3) {
                if (j<lenXNew-2) {
                        lineX[j]=lgcX[i];
                        lineY[j]=lgcY[i];
                }else{
                        // The idea here is to ensure that the last
set of coordinates are the same in both
                        // the new and old arrays
                        lineX[j]=lgcX[lenXOld-1];
                        lineY[j]=lgcY[lenXOld-1];
                }
                j++;
        }
        Plot.add("line", lineX, lineY);
        selectWindow(imnamePP);
        close("*");

        open(dirProcWCS+imnamePP);
                        // 6) Open new instance of merge image for
neurite ID
        selectWindow(imnamePP);
        setLocation(x_iw, y_iw);
        run("In [+]");
        run("In [+]");
        run("In [+]");
        run("In [+]");
        run("Scale to Fit");
        makeRectangle(x_new, 0, w_new, h_i);
        run("Crop");
        run("Flip Vertically");
        makeSelection( "polyline", lineX, lineY);
        setTool("arrow");
        waitForUser("Check Trace", "Verify that the line is correct
and spline fit is checked.\nMake adjustments as necessary before
proceeding.");
        roiManager("Add");
        roiManager("save", dirROIs+"/"+imnameBASE+".zip");
        selectWindow(imnamePP);
        close("*");
```

```
        // Make processed image of neurite for display purposes
        open(dirWCS+imnamesWC[0]);
        selectWindow(imnamesWC[0]);
        setLocation(x_iw, y_iw);
        selectWindow(imnamesWC[0]);
        run("In [+]");
        run("In [+]");
        run("Scale to Fit");
        run("Set Scale...", "distance=1 known="+cal+" pixel=1 unit=um
global");

        // NEED TO CROP THE ORIGINAL IMAGE IN ORDER FOR THE
COORDINATES OF THE NEURITE
        //      SELECTION TO MAKE SENSE
        makeRectangle(x_new, 0, w_new, h_i);
        run("Crop");
        // Have to flip the image because of a discrepancy between the
LGC coordinate system defination and the images'
        run("Flip Vertically");
        roiManager("Select", 0);
        run("Straighten...", "title="+imnamesWC[0]+"
line="+linewidth+" process");
        selectWindow(imnamesWC[0]);       // This closes the original
image while leaving the new straightened image open
        close();
        setMinAndMax(0, 65);
        run("Subtract Background...", "rolling=30 separate sliding");
        run("Invert");
        width = getWidth();
        if (width < 500) {
                newwidth = 500;
        }else {
                newwidth = width;
        }
        run("Canvas Size...", "width="+width+" height=75 position=Top-
Left");
        run("Scale Bar...", "width=5 height=4 font=20 color=White
background=None location=[Lower Right] hide");
        fnProc = imnameBASE+".ppJN.jpg";
        saveAs("Jpeg",dirProcJN+fnProc);
        close();

        // Make raw but straightened views of neurite, substrate, and
culture conditions
        //  Open images for all channels
        for( j = 0; j < channels.length ; j++) {
                file = dirWCS+imnamesWC[j];
                open(file);
        }
        run("Images to Stack", "name=stack" + " title=[] use");
```

```
        setLocation(x_iw, y_iw);
        run("Set Scale...", "distance=1 known="+cal+" pixel=1 unit=um
global");
        // NEED TO CROP THE ORIGINAL IMAGE IN ORDER FOR THE
COORDINATES OF THE NEURITE
        //      SELECTION TO MAKE SENSE
        makeRectangle(x_new, 0, w_new, h_i);
        run("Crop");
        run("Flip Vertically");
        roiManager("Select", 0);
        run("Straighten...", "title=stack line="+linewidth+"
process");

        fnCropped = imnameBASE+".JN";
        saveinfo = "["+dirNS+"]";
        run("Image Sequence... ", "format=TIFF name="+fnCropped+"
digits=1 save="+saveinfo);
        run("Close All");

        //fixInfo = [roiCB, roiNeu, fixCellBody, fixNeu];
        //fix1 = Array.concat(roiCB, roiNeu);
        //fix2 = Array.concat(fixCellBody,fixNeu);
        //fixInfo = Array.concat(array1,array2);
        return fixInfo;

}

function fixCheck(message){
// CHECK IF CURRENT THRESHOLDING IS SUFFICIENT OR IF USER NEEDS TO FIX
IT
        optFix = Array.concat("SKIP","FIX");
        Dialog.create("FIX THRESHOLDING");
        Dialog.addMessage(message);
        Dialog.addChoice("Do you want to manually fix the IDd blobs?",
optFix);
        Dialog.setLocation(x_d,y_d);
        Dialog.show();
        choiceFix = Dialog.getChoice();
        if (choiceFix == "FIX") {
                setForegroundColor(255, 252, 255);
                setTool("Paintbrush Tool");
                waitForUser("Use white paintbrush to make cuts\n"+
                        "then press enter.");
                setTool("Paintbrush Tool");
                setForegroundColor(0, 0, 0);
                waitForUser("Use black paintbrush to fill in gaps\n"+
                        "then press enter.");
        }
        return choiceFix;
}
```

```
function makeMontage(dirMon, dirProcJN, prepID, nsID){
        fnMontage = prepID+".Montage.AllCropped."+nsID+".jpg";
        imlist = getFileList(dirProcJN);
        for (i = 0; i < lengthOf(imlist); i++) {
                open(dirProcJN+imlist[i]);
        }
        rows = round(lengthOf(imlist)/2);
        run("Images to Stack", "method=[Copy (center)] fnMontage
title=[] use");
        setLocation(x_iw, y_iw);
        run("Make Montage...", "columns=2 rows="+toString(rows)+"
scale=0.50 label");
        saveAs("Jpeg",dirMon+fnMontage);
}
```

Appendix C

Python code for automated quantification of MEC-4 subcellular localization

C.1 Analysis details

The code presented in this Appendix uses the straightened neurite images generated as described in Appendix B to quantify the subcellular localization of MEC-4 along the neurite, identified as "peaks" in a one-dimensional signal. The code produces an Excel workbook that contains all of the measured values for every puncta of every TRN analyzed. This code was refined from the initial PeakFinder code written by A.Das, with necessary adjustments being made to accommodate images from *in vitro* rather than *in vivo* experiments.

C.2 PeakFinder.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
PACC_PeakFinder_v4.py

v4 switchs order of identifying peaks. Previous version of the code
was
calculating the threshold off of the raw signal but this was the
incorrect
approach since the background subtracted signal was the one being used
in find_peaks. Here:
    1. Calculate background
    2. Calculate raw neurite signal
    3. Calculate background subtracted raw neurite signal
    4. Use signal from step 3 to calculate threshold according to this
algo:
            1. Q3 is calculated for all pixel raw intensities
            2. A subset of pixels with intensity [0,Q3) is generated
            3. The mean and standard deviation is calculated for this
new subset
            4. The threshold for peak intensity is then the
mean+2*sigma.
                => This value is very close to the std for the full
population &
                guarantees an SNR of at least 1.
    5. Feed these parameters into find_peaks

Also made some changes to how the indices for the neurite versus
background
were calculated.

15MAR2020 - Changing Prominence to be calculated based off total
population std
"""
#LIBRARIES
import os
import matplotlib.pyplot as plt
import pandas
import glob
import datetime
import imageio
import numpy as np
from scipy.signal import find_peaks
from matplotlib import colors

# *** UPDATE NOTE TO STORE WITH ANALYSIS RUN ***
# Specify notes about this analysis run
rdmeNote = ["Red channel analysis\n"+
            "Added documentation of max normalized punctum intensity"]
# Specify color to analyze (green = 1, red = 0)
```

```
CH = 0
# Specify color file (green = 0, red =1 <- I realize how annoying this
is)
chExt = 'JN1'


# *** GET TIME OF ANALYSIS START ***
toa = str(datetime.datetime.today()).split()
# Analysis runs are saved with unique timestamps
today = toa[0]
now = toa[1]
timestamp = today.replace('-','')+'-'+now.replace(':','')[:6]

# *** WHAT TO ANALYZE ***
prepID = 'CCP_138'
# Cell-culture prep from where the NS arises
nsID = 'NS_02.01'
# Neurite set (NS) to analyze

#       ****       WHERE TO GET DATA & METADATA     ****
dirCCPs = "/Users/nerdette/Google Drive/Research/WormSense/Data/CCPs/"
# Location where data is stored for all preps
dirPrep = dirCCPs+prepID+"/"
# Location where all prep-specific data is stored
dirMetaD = dirPrep+"Metadata/"
# Directory for all prep-specific metadata
dirNS = dirPrep+"Images/Cropped/"+nsID+"/"
# Directory for NS images to be analyzed (req'd for analysis)
fnMDim = prepID+'.MetaD.IM.csv'
# Path to imaging metadata (req'd for analysis)
fnMDns = prepID+".MetaD."+nsID+".csv"
# Name of NS metatadata file (req'd for analysis)

#       ****        WHERE TO STORE RESULTS         ****
pathRes = dirPrep+"Analysis/"+timestamp+'/'
os.mkdir(pathRes)                                                    #
Output locataion for final excel workbook
fnRes = 'PACC_PFAnalysis.'+prepID+'.'+nsID+'.'+timestamp+'.xlsx'
# Output Excel file


#      ***   OUTPUT TEXT FILE TO DESCRIBE ANALYSIS RUN
os.chdir(pathRes)
rdmeFile = open(prepID+"_AnalysisRDME."+timestamp+".txt","w+")
rdmeFile.writelines(rdmeNote)
rdmeFile.close() #to change file access modes


#       ***       IMPORT METADATA FOR ANALYSIS      ***
dfMDim = pandas.read_csv(dirMetaD+fnMDim)
```

```
# Import original imaging metadata
dfMDim.columns = dfMDim.columns.str.strip().str.lower().str.replace('
',
                '_').str.replace('(', '').str.replace(')', '')
dfMDns = pandas.read_csv(dirMetaD+fnMDns)
# Import neurite set metadata
dfMDns.columns = dfMDns.columns.str.strip().str.lower().str.replace('
',
                '_').str.replace('(', '').str.replace(')', '')

#       ***     COUNT EXCLUSION INSTANCES AND TYPES     ***
dfExclusions = dfMDns['exclusion_reason'].value_counts()
# Poss entries are: None, Bipolar, Psuedo Bipolar, No neurites, other

#       ***         GET LIST OF IMAGES TO ANALYZE       ***
os.chdir(dirNS)
ims = glob.glob('*.'+chExt+'.tif')
# This line is for analyzing CH3 (green) images (.JNO images)

# *** ANALYSIS PARAMETERS ***
f = os.path.basename(__file__)
# Store filename of *.py analysis code
muperpx = dfMDim.loc[0,'calibration_um/pix']
# Get um/pix conversion factor from metadata


# *** CALCULATE PIXELS TO SAMPLE ***
pxTot = dfMDns.loc[0,'line_width']
pxBgndSize = (pxTot/4)
pxNeuSize = int(round(1/muperpx))
pxNeuStart = (pxTot-pxNeuSize)/2
# Only preliminary data was acquired at lower resolution. All images
used
# for official analysis should be taken at muperpx == .126.
# Analysis for .252 case is for backwards compatability.
if muperpx == .252:
    #calc indices for pixels to sample in image
    inB1 = 0
#Getting pixel indices
    inB2 = pxBgndSize-1
    inN1 = pxNeuStart
    inN2 = pxNeuStart+pxNeuSize-1
    inB3 = pxTot-pxBgndSize
    inB4 = pxTot-1
#Pixel index is size-1
elif muperpx == .126:
    #calc indices for pixels to sample in image
    inB1 = 0
#Getting pixel indices
    inB2 = pxBgndSize-1
```

```
    inN1 = pxNeuStart
    inN2 = pxNeuStart+pxNeuSize-1
    inB3 = pxTot-pxBgndSize
    inB4 = pxTot-1

pxN = [inN1,inN2]
#Pixel range for neurite
pxB = [inB1,inB2,inB3,inB4]
                                              #Pixel range for
background
dfPixInd = pandas.DataFrame(np.array([['background_1',inB1, inB2],
                                      ['neurite',inN1,inN2],
                                      ['background_2',inB3,inB4]]),
                            columns =
['region','index_start','index_end'])

#           *** SETUP DATAFRAMES TO SAVE AS SPREADSHEETS IN ***
#           ***     EXCEL FILE AT THE END OF ANALYSIS       ***
#General data about each image column
colsData =     ['date','image_id','prep_id','strain','ns_id',
                'tiv','pattern_geom','surface_proteins',
                'distance','normalized_distance',

'raw_intensity','background_intensity','neurite_intensity',
                'avg_norm_neu_int','max_norm_neu_int']

#Info about peaks found in image
colsPeaks =    ['date','image_id','prep_id','strain','ns_id',
                'tiv','pattern_geom','surface_proteins',
                'distance','normalized_distance',

'punctum_max_intensity','norm_punctum_max_int','punctum_width']

#Info on inter-puncta distances
colsIPDs =     ['date','image_id','prep_id','strain','ns_id',
                'tiv','pattern_geom','surface_proteins',
                'distance','normalized_distance',
                'inter-punctum_interval']

#Calculated info about image, peaks, and IPDs
colsAnalysis = ['date','image_id','prep_id','strain','ns_id',
                'tiv','pattern_geom','surface_proteins',

'image_size','max_neurite_length','average_neurite_intensity',
                'total_peaks','average_peaks_per_micron',
                'average_peak_intensity', 'average_peak_width',
                'average_ipd','median_ipd',
                'qTh','ss_mean','ss_median','ss_std','ss_n',
                'min_height','prominence']
```

```
#Track pixels used for each region of analysis
colsPixRange = ['region','starting_index','final_index']

#***initialize dataframes***
dfData = pandas.DataFrame()
dfPeaks = pandas.DataFrame()
dfIPDs = pandas.DataFrame()
dfAnalysis = pandas.DataFrame()


# *** MAIN PEAK FINDER ANALYSIS LOOP ***
os.chdir(dirNS)

for x in ims:
    # fileIm is the name of the original, regional image as it's
called in the
    #   MetaD.IM.csv file
    fileIm = x.split(".C")[0]

    # Import image and store it in a list of lists
    img = imageio.imread(x)[:,:,CH]
#CH should be an integer to specify which color images to analyze (1 =
GRN, 0 = RED)

    # Find index of fileIm in MetaD.IM.csv dataframe
    # Returns '0' if the file doesn't exist
    chk4file = dfMDim[dfMDim['image_name']==fileIm].index

    # Throw error if the image isn't found
    if(chk4file.size == 0):
        print("Error. Image file " +fileIm+ " is not in the original",
              " metadata sheet & will not be included in analysis.")
    # If image does exist, proceed with metadata gathering & data
analysis
    else:
        # ADD IMPORTANT METADATA TO MEASUREMENTS
        # Use index to get relavent metadata from MetaD.IM.csv
dataframe
        index = chk4file[0]
        date = dfMDim.at[index,'acquisition_date']
        strain = dfMDim.at[index,'strain']
        tiv = dfMDim.at[index,'tiv']
        pattern_geom = dfMDim.at[index,'pattern_geom']
        surface_proteins = dfMDim.at[index,'surface_proteins']

        # ADD IMAGE DATA TO DATA FRAME
        # Calculate image size
        imsize = np.shape(img)
        #***setup horizontal axes to represent image columns**
        d=np.arange(imsize[1])
```

```
#Array of integers to represent pixels along image (aka image columns)
        dist = d*muperpx
#Generate array of physical distances along image based on um:pix
conversion factor
        normdist=dist/dist[-1]
#Create a normalized axis to represent positions along image as 0->1
        #***break image up into background and neurite***
        n = img[pxN[0]:pxN[1], 0:]
#Extract neurite rows
        bg = np.concatenate((img[pxB[0]:pxB[1], 0:],
#Extract background rows
                             img[pxB[2]:pxB[3], 0:]))
        #***calculate values for analysis***
        rawf = np.mean(n, axis=0)
#Average raw neurite fluorescence
        bgf = np.mean(bg, axis=0)
#Average raw background fluorescence
        nf = rawf - bgf
#Background subtracted neurite fluorescence
        annf= nf/np.mean(nf,axis=0)
#Average normalized neurite fluorescencece
        mnnf = nf/np.amax(nf,axis=0)
        #***set negative nf values to zero
        for i in range(len(nf)):
            if nf[i]<0: nf[i]=0
        #***add image data dataframme***
        alldata1 = pandas.DataFrame({'date':[date]*imsize[1],
                                     'image_id':[x]*imsize[1],
                                     'prep_id':[prepID]*imsize[1],
                                     'strain':[strain]*imsize[1],
                                     'ns_id':[nsID]*imsize[1],
                                     'tiv':[tiv]*imsize[1],
                                     'pattern_geom':
[pattern_geom]*imsize[1],
                                     'surface_proteins':
[surface_proteins]*imsize[1],
                                     'distance':dist,
                                     'normalized_distance':normdist,
                                     'raw_intensity':rawf,
                                     'background_intensity':bgf,
                                     'neurite_intensity':nf,
                                     'avg_norm_neu_int':annf,
                                     'max_norm_neu_int':mnnf},
columns=colsData)
        dfData=dfData.append(alldata1)


        # DETERMINE PEAK LOCATIONS
        # Calculate minimum height and prominence values
        # 1. Calculate Third Quartile for all image neurite pixels
```

```
rawf values
        descNF= alldata1['neurite_intensity'].describe()
        qTh = descNF[['75%']][0]
        # 2. Create a subset of pixels that are below the third
quartile
        dfSS_alldata1 = alldata1[alldata1['neurite_intensity'] < qTh]
        # 3. Calculate important stats for this subset
        descSS = dfSS_alldata1['neurite_intensity'].describe()
        mean = descSS[['mean']][0]
        median = descSS[['50%']][0]
        stdSS = descSS[['std']][0]
        n = descSS[['count']][0]
        # 4. Use these statistics to calcualte cutoff values
        minHeight = mean+2*stdSS
        #prom = 2*std
        # 5. Use pop std to calculate prominence cutoff values
        descNI = alldata1['neurite_intensity'].describe()
        prom = descNI[['std']][0]


        #***find peaks***
        peaks = find_peaks(nf, height=minHeight, prominence=prom,
#Relheight is used to calculate peak width, it is a % of peak
prominence
                           width=0, rel_height=0.5)
        pd = peaks[0]*muperpx
#Convert pixel distances to physical distances
        pnd = pd/dist[-1]
#Calculated normalized physical distances (0->1)
        pmi = [nf[i] for i in peaks[0]]
#Get intensity for each punctum location
        pmi_norm = [mnnf[i] for i in peaks[0]]
#Get max normalized intensity for each punctum location
        #***calculate punctum spacing***
        ipd = np.diff(pd)
#Here, .diff() returns physical distance between puncta
        ipdd = [pd[i]+ipd[i]/2 for i in range(0,len(ipd))]
#Inter-punctum interval:
        ipdnd = ipdd/dist[-1]
        #***add peak data to dataframe***
        alldata2 = pandas.DataFrame({'date':[date]*len(pd),
                                     'image_id':[x]*len(pd),
                                     'prep_id':[prepID]*len(pd),
                                     'strain':[strain]*len(pd),
                                     'ns_id':[nsID]*len(pd),
                                     'tiv':[tiv]*len(pd),
                                     'pattern_geom':
[pattern_geom]*len(pd),
                                     'surface_proteins':
[surface_proteins]*len(pd),
```

```
                                    'distance':pd,
                                    'normalized_distance':pnd,
                                    'punctum_max_intensity':pmi,
                                    'norm_punctum_max_int':pmi_norm,
                                    'punctum_width':peaks[1]
['widths']*muperpx},
                                   columns=colsPeaks)
        dfPeaks=dfPeaks.append(alldata2)
        #***add Inter-punctum data to dataframe***
        alldata3 = pandas.DataFrame({'date':[date]*len(ipd),
                                    'image_id':[x]*len(ipd),
                                    'prep_id':[prepID]*len(ipd),
                                    'strain':[strain]*len(ipd),
                                    'ns_id':[nsID]*len(ipd),
                                    'tiv':[tiv]*len(ipd),
                                    'pattern_geom':
[pattern_geom]*len(ipd),
                                    'surface_proteins':
[surface_proteins]*len(ipd),
                                    'distance':ipdd,
                                    'normalized_distance':ipdnd,
                                    'inter-punctum_interval':ipd},
                                   columns=colsIPDs)
        dfIPDs=dfIPDs.append(alldata3)

        #***add analysis to dataframe***
        # calculated info about image, peaks, and IPDs
        frame = pandas.DataFrame([[date, x, prepID, strain, nsID, tiv,
# 'Date','ImageID','Prep ID','Strain','NS ID','tiv'
                                   pattern_geom, surface_proteins,
# 'pattern_geom','surface_proteins',
                                   imsize[1], dist[-1], np.mean(nf),
# 'Image size','Max neurite length','Average neurite intensity'
                                   len(pd), len(pd)/dist[-1],
# 'Total peaks', 'Average Peaks per Micron'
                                   np.mean(pmi),
# 'Average peak intensity'
                                   np.mean(peaks[1]
['widths']*muperpx),       # 'Average peak width'
                                   np.mean(ipd), np.median(ipd),

qTh,mean,median,stdSS,n,minHeight,prom]],            # 'Average
ipd','Median ipd'
                                   columns=colsAnalysis)
        dfAnalysis = dfAnalysis.append(frame)

        #   ***     Create image specific analysis file    ****

        # Set main figure properties
        SMALL_SIZE = 4
```

```
        MEDIUM_SIZE = 6
        BIGGER_SIZE = 8
        plt.style.use('dark_background')
        plt.rc('font', size=MEDIUM_SIZE)          # controls default
text sizes
        plt.rc('axes', titlesize=MEDIUM_SIZE)     # fontsize of the
axes title
        plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x
and y labels
        plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the
tick labels
        plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the
tick labels
        plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
        plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the
figure title
        fig, axes = plt.subplots(3, 2, dpi=300,figsize=(6.5,4.5))
        ((ax1, ax2),(ax3,ax4),(ax5,ax6)) = axes
        fig.suptitle(x+'\nAnalysis '+timestamp+'\nProminence =
'+str(round(prom,2))+' Min Height = '+str(round(minHeight)))
        plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=.5,
wspace=0.4)

        # Plot #1 is the raw image
        ax1 = plt.subplot(311)
#subplot(nrows, ncols, index, **kwargs)
        ax1.set_title('Raw Image')
        # Shift the plot down so that it doesn't overlap with the
title
        box = ax1.get_position()
        box.y0 = box.y0 - 0.1
        box.y1 = box.y1 - 0.1
        ax1.set_position(box)
        ax1.set_ylim((0,pxTot))
        ax1.set_yticks((pxBgndSize/2,pxTot/2,pxTot-(pxBgndSize/2)))
        labels = ['Background','Neurite','Background']
        ax1.yaxis.set_ticklabels(labels,position=(0,.05))
        ax1.set_xlabel('Pixel Index')
        ax1.hlines((inB2,inN1,inN2,inB3),0,imsize[1],color='w',
linewidth =.2,linestyles= 'dashed')
        ax1.imshow(img, vmin=0,vmax=180)

        # Plot #2 is the histogram of pixel intensities
        ax3 = plt.subplot(323)
        ax3.set_title('Distribution of Pixel Intensities')
        ax3.set_xlabel('Pixel Intensity (AU)')
        ax3.set_ylabel('Pixel Count')
        n_bins = 25
        N, bins, patches = ax3.hist(nf, bins=n_bins)
        ax3.set_ylim((0,N.max()*1.1))
```

```
        p1 = plt.vlines(qTh, 0, N.max()*1.1, color='w',linewidth=.25)
        p2 = plt.vlines(minHeight, 0, N.max()*1.1,
color='m',linewidth=.25)
        p3 = plt.vlines(prom, 0, N.max()*1.1, color='b',linewidth=.25)
        # N is the count in each bin, bins is the lower-limit of the
bin
        # We'll color code by height, but you could use any scalar
        fracs = N / N.max()
        # we need to normalize the data to 0..1 for the full range of
the colormap
        norm = colors.Normalize(fracs.min(), fracs.max())
        # Now, we'll loop through our objects and set the color of
each accordingly
        for thisfrac, thispatch in zip(fracs, patches):
            color = plt.cm.viridis(norm(thisfrac))
            thispatch.set_facecolor(color)
        plt.legend((p1,p2,p3),
                   ('Third Quartile', 'Minimum Height','Prominence'),
loc='upper right', bbox_to_anchor=(1,1))

        # Plot #3 is the raw background and neurite signal
        ax4 = plt.subplot(324)
        ax4.set_title('Mean Pixel Intensity for Region of Interest')
        ax4.set_xlabel('Distance (um)')
        ax4.set_ylabel('Intensity (AU)')
        ax4.set_xlim(0,max(dist))
        ax4.set_ylim(0,max(rawf)*1.1)
        plt.yticks([0,round((max(rawf)*1.1)/
2,-1),round((max(rawf)*1.1),-1)])
        p4 = plt.plot(dist, bgf, 'c-')
        p5 = plt.plot(dist, rawf, 'g-')
        plt.legend((p4[0], p5[0]),
                   ('Background', 'Neurite'),  loc='best',
bbox_to_anchor=(1,1))

        #Plot #4 is the corrected neurite signal with identified peaks
        ax5 = plt.subplot(313)
        ax5.set_title('Identified Peaks')
        plt.xlabel('Distance (um)')
        plt.ylabel('Intensity (AU)')
        ax5.set_ylim(0,max(nf)*1.1)
        ax5.set_xlim(0,max(dist))
        plt.yticks([0,round((max(nf)*1.1)/
2,-1),round((max(nf)*1.1),-1)])
        p6 = plt.plot(dist, np.array([minHeight for i in
xrange(len(dist))]), 'm-')
        p7 = plt.plot(dist, nf, 'b-')
        p8 = plt.plot(pd, pmi, 'mo',markersize=4)
        plt.legend((p6[0],p7[0], p8[0]), ('Minimum Height',
                    'Corrected Neurite Signal', 'Identified Peak'),
```

```
                    loc='upper right', bbox_to_anchor=(1,1))

        plt.savefig(pathRes+x[:-4]+'_pf.png',  bbox_inches='tight',
                    dpi = 300,
                    format = "png")

        plt.close()

    #END ANALYSIS#

#***store user-specified and analysis parameters***
dfParameters = pandas.DataFrame(data={'1. Date of analysis':today,
                                      '2. Time of analysis':now,
                                      '3. Microns per pixel':muperpx,
                                      '4. Script name':f},
                                       index=[0])

#OUTPUT DATAFRAMES AS SHEETS IN EXCEL FILE
wb = pandas.ExcelWriter(pathRes+fnRes, engine='xlsxwriter')
dfData.to_excel(wb, sheet_name='Data')
dfPeaks.to_excel(wb, sheet_name='Peaks')
dfIPDs.to_excel(wb, sheet_name='IPDs')
dfAnalysis.to_excel(wb, sheet_name='Analysis')
dfExclusions.to_excel(wb, sheet_name='Exclusions')
dfPixInd.to_excel(wb,sheet_name='Pixel Indices')
dfParameters.to_excel(wb, sheet_name='Parameters')
wb.save()
```

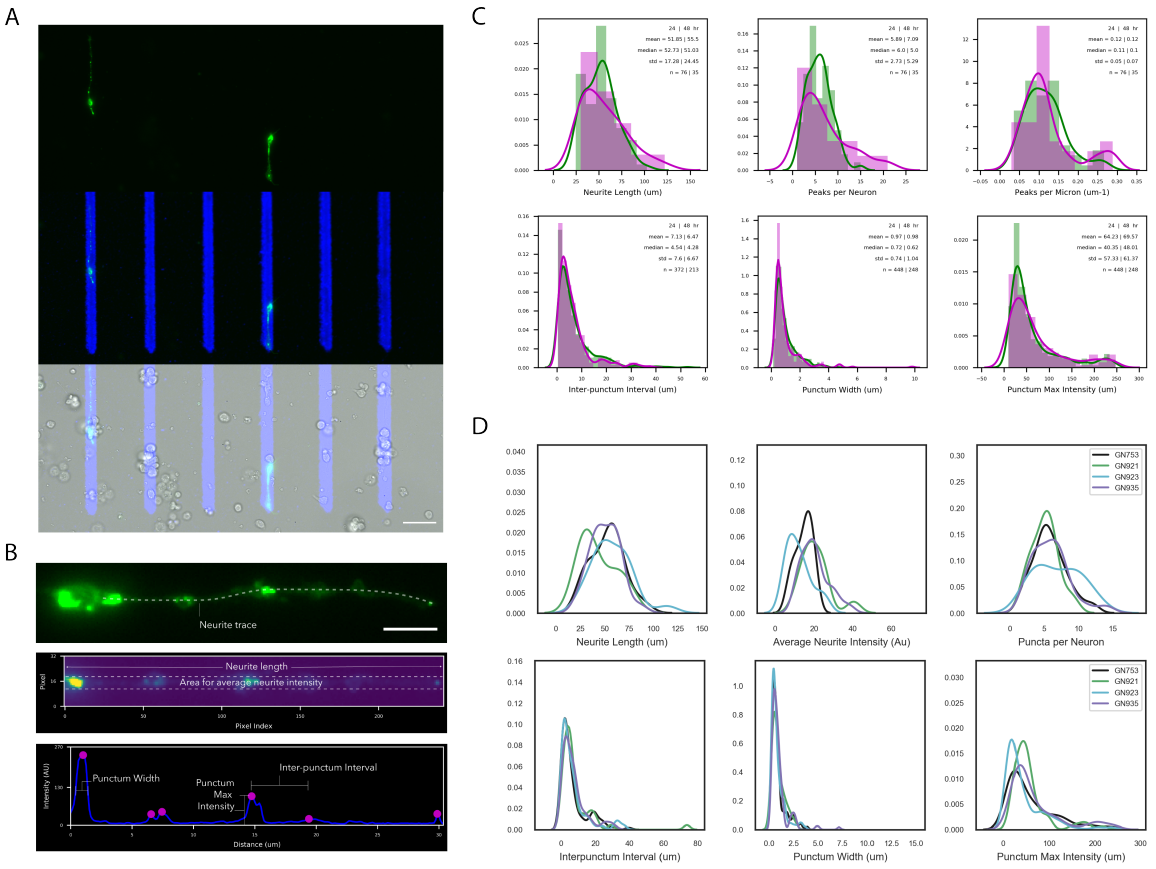


Figure C.1: **Isolated TRNs possess fewer mNG::MEC-4 puncta that *in vivo* correlates**. A) Representative widefield images of mNG::MEC-4 expressing TRNs, isolated from *C. elegans* embryos, at 24 hrs in culture (top), overlay of TRNs with fluorescent PNA micropatterns (middle), and overlay of all both fluorescent channels with bright-field (bottom). Scale bar is 20 µm. B) Annotated images of mNG::MEC-4 expressing TRN showing line traced for generation of just the neurite image (top) and intermediate data produced during analysis (middle and bottom). Peak Finder algorithm produces the middle and bottom plots for each image analyzed. Blue trace represents the reduced dimension fluorescence intensity plot for the neurite region outlined in the middle panel following background subtraction. Pink dots represent peaks identified by the algorithm. Scale bar in top panel is 5 µm. C) Peak Finder results for neurites shown in Figure 4.2.B, presented as histograms with kernel density estimates (KDE) for images analyzed at 24 hrs (green) and 48 hrs (magenta) in culture. D) KDEs for isolated TRNs from parent strains expressing both mNG::MEC-4 and mutations to genes encoding ECM proteins (GN921 - *mec-1(e1496)*, GN923 - *nid-1(cg119)*, GN935 - *mec-9(u437)*) overlaid with mNG::MEC-4 expressing strain in wild type background (GN753).

Appendix D

Python code for automated colocalization analysis of MEC-4 and RAB-3

D.1 Analysis details

The code presented in this Appendix uses the Excel files generated in Appendix C to quantify the degree of colocalization between MEC-4 and RAB-3 puncta. To do this is calculates the Pearson Correlation Coefficient for the two signals around the MEC-4 puncta as determined in PeakFinder. This approach uses the second derivative of the two signals to quantify the degree to which these signals are synchronized between the two signals and assigns a value from -1 (anti-correlated) to 1 (correlated). It is agnostic to signal intensity and is only concerned with whether the signals are increasing, plateauing, or decreasing together.

D.2 PeakFinder.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
PACC_PeakCorr_v1.py
Created on Mon Mar 15 14:47:28 2021
@author: nerdette

The purpose of this code is to calculate the Pearson Correlation
Coefficients
for the 1 µm regions centered at each peak in the green channel as
related
to same 1 µm region in the red channel. We have seen that the average
punctum
width in the green channel is about 1 µm, which equates to 8 pixels,
and
should be enough to account for the potential chromatic shift.

Currently:
    - Neurite intensities are normalized to the maximum intensity
        -> This is so that everything is between 0 and 1
    - The cutoff for peak colocalization will be a pearson correlation
      coefficient of .25
    - Only considering peaks with punctum width >= 3 pixels = .378


"""

#                           *** LIBRARIES ***
import pandas as pd
import os
import datetime
from shutil import copyfile
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import imageio
from matplotlib import colors


#                       *** SET UP FIGURE PROPERTIES  ***
SMALL_SIZE = 4
MEDIUM_SIZE = 6
BIGGER_SIZE = 8
#plt.style.use('dark_background')
plt.rc('font', size=MEDIUM_SIZE)          # controls default text
sizes
plt.rc('axes', titlesize=MEDIUM_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y
```

```
labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure
title

#                   *** GET TIME OF ANALYSIS START ***
toa = str(datetime.datetime.today()).split()
#Time of analysis
today = toa[0]
now = toa[1]
timestamp = today.replace('-','')+'-'+now.replace(':','')[:6]

#                         *** WHAT TO ANALYZE ***
prepID = 'CCP_138'
nsID = 'NS_02.01'
ana2plot = ['20210316-140029','20210316-140206']
#Input analysis codes for mNG, mCH
muperpx = .126
#Microns to pixels calibration
minPW = 3*muperpx
pxIndRange = 4
#Number of pixels needing to be sampled around peak


#                        **** WHERE TO GET DATA ****
dirMain = "/Users/nerdette/Google Drive/"
dirCode = dirMain+"Code/PACC/"
dirCCPs = dirMain+"Research/WormSense/Data/CCPs/"  # Location where
data is stored for all preps
dirPrep = dirCCPs+prepID+"/"
# Location where all prep-specific data is stored
dirNS = dirPrep+"Images/Cropped/"+nsID+"/"
# Directory for NS images to be analyzed (req'd for analysis)
fnAna = []
for entry in ana2plot:

fnAna.append(dirPrep+"Analysis/"+entry+'/'+'PACC_PFAnalysis.'+prepID+'
.'+nsID+'.'+entry+'.xlsx')               # xlsx file containing data
to analyze


#                        **** WHERE TO STORE DATA ****
dirSV=dirPrep+'Results/PeakCorr.'+timestamp+'/'
# Folder for storing output files
fnPNG = 'PC.'+timestamp+'.svg'
fnXLSX= 'PC.'+timestamp+'.xlsx'

#               **** MAKE DIRECTORY FOR STORING PLOTTING RESULTS ****
```

```
os.mkdir(dirSV)

#                       *** SAVE A COPY OF THIS FILE ***
f = os.path.basename(__file__)
copyfile(dirCode+f, dirSV+f)

#               **** LOAD THE DATA & BUILD THE DFs ****

# Initialize data frames for storing and exporting analysis
dfDataPS = pd.DataFrame()
# For storing data for range of pixels around the peak
dfPeaksAll = pd.DataFrame()
# For storing data associated with just the peak

# Import data
xfDG = pd.ExcelFile(fnAna[0])
# Load the XLSX workbook for green channel
xfDR = pd.ExcelFile(fnAna[1])
# Load the XLSX workbook for red channel
dfDataG = pd.read_excel(xfDG, 'Data')
# Load the Data spreadsheet
dfDataR = pd.read_excel(xfDR, 'Data')
# Load the Data spreadsheet
dfPeaks = pd.read_excel(xfDG, 'Peaks')
# Load the Peaks spreadsheet
dfAnalysis = pd.read_excel(xfDG, 'Analysis')
# Load the Analysis spreadsheet
# Get list of all images in analysis
# Compile to fix funny column names and indices
data = [dfDataG,dfDataR,dfPeaks,dfAnalysis]
for df in data:
# Fix funny column names and indexing***
    df.columns = df.columns.str.strip().str.lower().str.replace(' ',
'_').str.replace('(', '').str.replace(')', '')     #Fixing messy
column names makes life easier
    df.reset_index(drop=True, inplace=True)

# Rename unnamed column
dfDataG.rename( columns={'unnamed:_0':'in_pix'}, inplace=True )
dfDataR.rename( columns={'unnamed:_0':'in_pix'}, inplace=True )

#Iterate through all of the images analyzed
#For every image analyzed,
for index, row in dfAnalysis.iterrows():
    imageG = row['image_id']
#The actual name of image that was originally analyzed in PeakFinder
    cellID = imageG.split(".J")[0]
#Unique cell ID that can be used to construct corresponding red
channel image name
    imageR = cellID+'.JN1.tif'
```

```
    totPeaks = row['total_peaks']

    # Import image and store it in a list of lists
    imgG = imageio.imread(dirNS+imageG)[:,:,1]
#CH should be an integer to specify which color images to analyze (1 =
GRN, 0 = RED)
    imgR = imageio.imread(dirNS+imageR)[:,:,0]

    #Only analyze the images that have peaks
    if totPeaks > 0 :
        # Get a list of indices for all the peaks for this image in
the Peaks data frame
        # These locations were determined using the green image so
that's the image_id we use to search
        indices = dfPeaks.index[dfPeaks['image_id'] ==
imageG].tolist()
        # For every peak associated with this image
        for i in indices:
            peakLoc = dfPeaks.at[i,'distance']
#Distance of peak (in um) from cell body
            peakWid = dfPeaks.at[i,'punctum_width']
#Full width half maximum of peak
            peakInt = dfPeaks.at[i,'norm_punctum_max_int']
#Maximum punctum intensity normalized to max neurite intensity
            #Restrict observation space to only puncta bigger than 3
pixels
            #if peakWid > minPW:
            #Need to calculate distance values or indices so I know
what range of rows to sample for calculating correlation coefficient
            # 1. Find the index of this peak in the Data df, needs to
match image_id
            # The images are the same but their ids are based on red
or green. We need to get the indices for the red image
            peakI_G = dfDataG[(dfDataG['image_id'] ==
imageG)&(dfDataG['distance'] == peakLoc)].index[0]
            peakI_R = dfDataR[(dfDataR['image_id'] ==
imageR)&(dfDataR['distance'] == peakLoc)].index[0]
            peakI_ImgG = dfDataG.at[peakI_G,'in_pix']

            # 2. Slice the Data df for the range around the peak
            startG = (peakI_G-pxIndRange)
            endG = (peakI_G+pxIndRange+1)
            startR = (peakI_R-pxIndRange)
            endR = (peakI_R+pxIndRange+1)
            dfSS_G = dfDataG.iloc[startG:endG]
            dfSS_R = dfDataR.iloc[startR:endR]

            # 3. Compile into a single df
            # Initialize the data frame filling values that won't
change
```

```
            dfDataSS = pd.DataFrame()
            for count in range(pxIndRange*2+1):
                dfDataSS = dfDataSS.append({'cell_id':cellID,
                                            'image_g':imageG,
                                            'image_r':imageR,
                                            },ignore_index=True)

            # Add select columns from dfSS for red and green
            dfDataSS['i_dfG'] = dfSS_G.index
# Track the indicies as they match to dfDataG
            dfDataSS['i_dfR'] = dfSS_R.index
# Track the indicies as they match to dfDataR
            dfDataSS['distG'] = dfSS_G['distance'].to_numpy()
            dfDataSS['distR'] = dfSS_R['distance'].to_numpy()
            dfDataSS['intensityG'] =
dfSS_G['max_norm_neu_int'].to_numpy()
            dfDataSS['intensityR'] =
dfSS_R['max_norm_neu_int'].to_numpy()

            # 4. Calculate the Pearson Correlation Coefficient for the
two channels' intensities
            col1 = dfDataSS['intensityG']
            col2 = dfDataSS['intensityR']
            pcc = round(col1.corr(col2),3)

            # 5. Store range data for export
            dfDataPS = dfDataPS.append(dfDataSS)

            # 6. Store the peak data for export
            dfPeaksAll = dfPeaksAll.append({'cell_id':cellID,
                                            'peak_no':i,
                                            'peak_loc_px':peakI_ImgG,
                                            'peak_loc_um':peakLoc,
                                            'peak_width':peakWid,
                                            'peak_npmi':peakInt,

'pearson_coeff':pcc},ignore_index=True)


        # Once all peaks have been analyzed, generate an output figure
to show analysis
        # Collect Peak Info
        x = dfDataG[ dfDataG['image_id' ]==imageG].distance.to_numpy()
#Distances in um for image (for plotting)
        yG =
dfDataG[ dfDataG['image_id' ]==imageG].max_norm_neu_int.to_numpy()
#Max normalized neurite intensity in mNG (au)
        yR =
dfDataR[ dfDataR['image_id' ]==imageR].max_norm_neu_int.to_numpy()
#Max normalized neurite intensity in mCh (au)
```

```
        peaksX =
dfPeaksAll[ dfPeaksAll['cell_id']==cellID ].peak_loc_um.to_numpy()
#Peak location (um)
        peaksY =
dfPeaksAll[ dfPeaksAll['cell_id']==cellID ].peak_npmi.to_numpy() #Peak
max normalized intensity (au)
        peaksR =
dfPeaksAll[ dfPeaksAll['cell_id']==cellID ].pearson_coeff.to_numpy()
#Pearson correlation coefficient for that peak
        # Generate text string for printing r values
        textStr = 'Peak No | Pearson Coefficient:\n'
        for k in range(len(peaksX)):
            textStr += str(k)+'           |'+str(peaksR[k])+'\n'

        fig, axes = plt.subplots(3, 2)#,tight_layout=True,
gridspec_kw={'height_ratios':[1,1,2.75]})
        #fig.subplots_adjust(bottom=0.3, wspace=0.5)

        ((ax1,ax2),(ax3,ax4),(ax5,ax6)) = axes
        fig.suptitle('mNG-mCh Peak Correlation Analysis\n'+'Cell ID:
'+cellID)
        #plt.subplots_adjust(top = 0.99, bottom=0.01, hspace=.5,
wspace=0.4)

        # Plot #1 is the raw image in the green channel
        ax1 = plt.subplot(321)
#subplot(nrows, ncols, index, **kwargs)
        ax1.set_title('Raw Image - mNG\n'+imageG+' | Analysis File
'+ana2plot[0])
        ax1.set_xlabel('Distance (pixels)')
        # Shift the plot down so that it doesn't overlap with the
title
        #box1 = ax1.get_position()
        #box1.y0 = box1.y0 - 0.05
        #box1.y1 = box1.y1 - 0.05
        #ax1.set_position(box1)
        ax1.imshow(imgG, vmin=0,vmax=180)

        #Plot#2 are the pearson coefficients
        ax2 = plt.subplot(122)
        ax2.set_title('Pearson Correlation Coefficients\nFor
Identified mNG Peaks')
        ax2.axis('off')
        # Shift the plot down so that it doesn't overlap with the
title
        box = ax2.get_position()
        box.y0 = box.y0 - 0.1
        box.y1 = box.y1 - 0.1
        ax2.set_position(box)
        boxText = ax2.get_position()
```

```
        boxText.y0 = boxText.y0 - .1
        boxText.y1 = boxText.y1 - .05
        plt.text(.1,boxText.y1,textStr)

        # Plot #3 is the raw image in the red channel
        ax3 = plt.subplot(323)
#subplot(nrows, ncols, index, **kwargs)
        ax3.set_title('Raw Image - mCH\n'+imageG+' | Analysis File
'+ana2plot[1])
        ax3.set_xlabel('Distance (pixels)')
        # Shift the plot down so that it doesn't overlap with the
title
        #box2 = ax2.get_position()
        #box2.y0 = box2.y0 - 0.1
        #box2.y1 = box2.y1 - 0.1
        #ax2.set_position(box2)
        ax3.imshow(imgR, vmin=0,vmax=180)

        # Plot #4 are the overlapping 1D intensity profiles for both
channels
        ax5 = plt.subplot(325)
        ax5.set_title('Normalized 1D Intensity Profiles')
        ax5.set_ylim((0,1.25))
        ax5.set_yticks((0,.25,.5,.75,1))
        ax5.set_xlim((0,max(x)))
        ax5.set_xlabel('Distance (um)')
        p1 = plt.plot(x,yG,'c-')
        p2 = plt.plot(x,yR,'m-')
        p3 = plt.plot(peaksX,peaksY,'bo',markersize=4)
        y = 0
        for x in peaksX:
            plt.text(x,peaksY[y]
+.05,str(y),horizontalalignment='center')
            y = y+1
        plt.legend((p1[0],p2[0], p3[0]), ('mNG::MEC-4',
                    'RAB-3::mCh', 'Identified Peak'),
                    loc='upper center',
                    bbox_to_anchor=(0.5, -0.4),fancybox=False,
shadow=False, ncol=3)
        plt.savefig(dirSV+cellID+'_pcc.svg',  bbox_inches='tight',
                    dpi = 300,
                    format = "svg")
        plt.close()
        # END ANALYSIS FOR THIS CELL
    #END ANALYSIS FOR DATA SET

# Once all images have been analyzed, plot all pearson coefficients
rAll = dfPeaksAll['pearson_coeff'].to_numpy()
mode = stats.mode(rAll)[0]
plt.suptitle('Distribution of Pearson Correlation Coefficients')
```

```
plt.xlabel('Pearson Correlation Coefficient')
plt.ylabel('Count')
n_bins = 20
N, bins, patches = plt.hist(rAll, bins=n_bins)
plt.ylim((0,N.max()*1.1))
plt.xlim((-1.1,1.1))
plt.xticks((-1,-.75,-.5,-.25,0,.25,.5,.75,1))
p0 = plt.vlines(mode, 0, N.max()*1.1, color='w',linewidth=.25)
#p2 = plt.vlines(minHeight, 0, N.max()*1.1, color='m',linewidth=.25)
#p3 = plt.vlines(prom, 0, N.max()*1.1, color='b',linewidth=.25)
# N is the count in each bin, bins is the lower-limit of the bin
# We'll color code by height, but you could use any scalar
fracs = N / N.max()
# we need to normalize the data to 0..1 for the full range of the
colormap
norm = colors.Normalize(fracs.min(), fracs.max())
# Now, we'll loop through our objects and set the color of each
accordingly
for thisfrac, thispatch in zip(fracs, patches):
    color = plt.cm.viridis(norm(thisfrac))
    thispatch.set_facecolor(color)
plt.text(mode[0],N.max()*.75,'Mode =
'+str(mode[0]),horizontalalignment='right')
plt.savefig(dirSV+fnPNG,  bbox_inches='tight',
                    dpi = 300,
                    format = "svg")

#OUTPUT DATAFRAMES AS SHEETS IN EXCEL FILE
wb = pd.ExcelWriter(dirSV+fnXLSX, engine='xlsxwriter')
dfPeaksAll.to_excel(wb, sheet_name='PeakAnalysis')
dfDataPS.to_excel(wb, sheet_name='AllData')
wb.save()
```

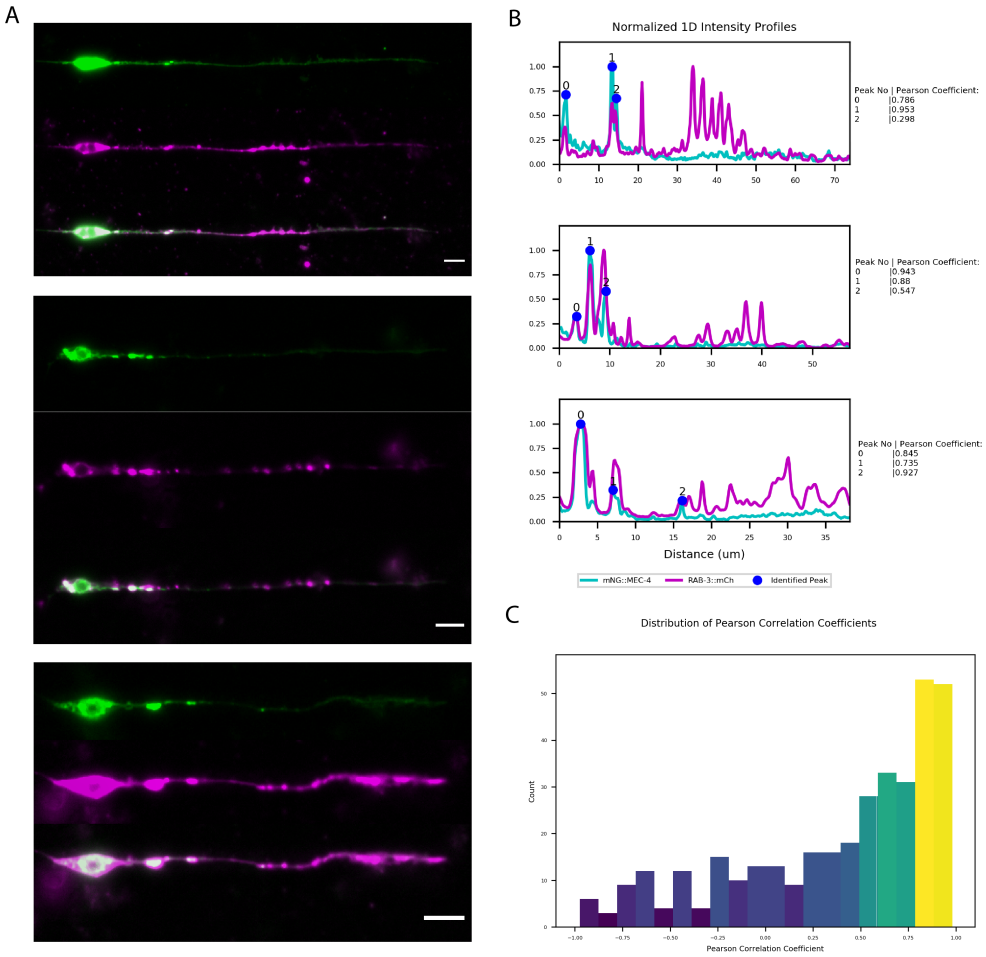


Figure D.1: **Majority of MEC-4 puncta colocalize with RAB-3 vesicle marker**. A) Representative images of isolated TRNs from parent strain expressing both the mNG::MEC-4 fusion protein (green) and a RAB-3::mCh vesicle marker (magenta; GN1016). Scale bar is 5 µm in all images. B) Quantification of Pearson Correlation Coefficients (PCC) between green and red intensity traces about the identified puncta for the three representative images shown in panel A, in the same vertical order. Cyan trace corresponds to the mNG::MEC-4 imaging channel, magenta trace to the RAB-3::mCh imaging channel, and blue dots represent mNG::MEC-4 peaks as identified by the Peak Finder algorithm. C) Pooled distribution of PCCs for 88 analyzed TRNs. For all PCCs, values correspond to anti-correlated (-1), no relation (0), or fully correlated (1).

Bibliography

[1] Ravi Das, Stefan Wieser, and Michael Krieg. Neuronal stretch reception–making sense of the mechanosense. *Experimental cell research*, 378(1):104–112, 2019.

[2] Seung-Hyun Woo, Viktor Lukacs, Joriene C De Nooij, Dasha Zaytseva, Connor R Criddle, Allain Francisco, Thomas M Jessell, Katherine A Wilkinson, and Ardem Patapoutian. Piezo2 is the principal mechanotransduction channel for proprioception. *Nature Neuroscience*, 18(12):1756–1762, 2015.

[3] Miriam B Goodman and Erich M Schwarz. Transducing touch in caenorhabditis elegans. *Annual review of physiology*, 65(1):429–452, 2003.

[4] Reza Sharif-Naeini. Contribution of mechanosensitive ion channels to somatosensation. *Progress in molecular biology and translational science*, 131:53–71, 2015.

[5] Adam P Christensen and David P Corey. Trp channels in mechanosensation: direct or indirect activation? *Nature Reviews Neuroscience*, 8(7):510–521, 2007.

[6] Stephen G Brohawn. How ion channels sense mechanical force: insights from mechanosensitive k2p channels traak, trek1, and trek2. *Annals of the New York Academy of Sciences*, 1352(1):20–32, 2015.

[7] JG McLeod. Investigation of peripheral neuropathy. *Journal of neurology, neurosurgery, and psychiatry*, 58(3):274, 1995.

[8] P Marchettini, M Lacerenza, E Mauri, and C Marangoni. Painful peripheral neuropathies. *Current neuropharmacology*, 4(3):175–181, 2006.

[9] Didier Bouhassira, Michel Lantéri-Minet, Nadine Attal, Bernard Laurent, and Chantal Touboul. Prevalence of chronic pain with neuropathic characteristics in the general population. *Pain*, 136(3):380–387, 2008.

[10] Yusuke Fukuda, Yihang Li, and Rosalind A Segal. A mechanistic understanding of axon degeneration in chemotherapy-induced peripheral neuropathy. *Frontiers in neuroscience*, 11:481, 2017.

[11] Raul Krauss, Todd Bosanac, Rajesh Devraj, Thomas Engber, and Robert O Hughes. Axons matter: the promise of treating neurodegenerative disorders by targeting sarm1-mediated axonal degeneration. *Trends in pharmacological sciences*, 41(4):281–293, 2020.

[12] Victoria E Abraira and David D Ginty. The sensory neurons of touch. *Neuron*, 79(4):618–639, 2013.

[13] Samata Katta, Michael Krieg, and Miriam B Goodman. Feeling force: physical and physiological principles enabling sensory mechanotransduction. *Annual review of cell and developmental biology*, 31:347–371, 2015.

[14] DM Bouley, CN Alarcon, T Hildebrandt, and Caitlin E O'Connell-Rodwell. The distribution, density and three-dimensional histomorphology of pacinian corpuscles in the foot of the asian elephant (elephas maximus) and their potential role in seismic communication. *Journal of Anatomy*, 211(4):428–435, 2007.

[15] Sean M Maguire, Christopher M Clark, John Nunnari, Jennifer K Pirri, and Mark J Alkema. The c. elegans touch response facilitates escape from predacious fungi. *Current Biology*, 21(15):1326–1330, 2011.

[16] William R Schafer. Mechanosensory molecules and circuits in c. elegans. *Pflügers Archiv-European Journal of Physiology*, 467(1):39–48, 2015.

[17] B Fritzsch and KW Beisel. Keeping sensory cells and evolving neurons to connect them to the brain: molecular conservation and novelties in vertebrate ear development. *Brain, behavior and evolution*, 64(3):182–197, 2004.

[18] Lise Frézal and Marie-Anne Félix. The natural history of model organisms: C. elegans outside the petri dish. *eLife*, 4:e05849, 2015.

[19] Adam L Nekimken, Eileen A Mazzochette, Miriam B Goodman, and Beth L Pruitt. Forces applied during classical touch assays for caenorhabditis elegans. *PloS one*, 12(5):e0178080, 2017.

[20] Bryan C Petzold, Sung-Jin Park, Eileen A Mazzochette, Miriam B Goodman, and Beth L Pruitt. Mems-based force-clamp analysis of the role of body stiffness in c. elegans touch sensation. *Integrative Biology*, 5(6):853–864, 2013.

[21] Robert O'Hagan, Martin Chalfie, and Miriam B Goodman. The mec-4 deg/enac channel of caenorhabditis elegans touch receptor neurons transduces mechanical signals. *Nature neuroscience*, 8(1):43–50, 2005.

[22] Martin Chalfie and John Sulston. Developmental genetics of the mechanosensory neurons of caenorhabditis elegans. *Developmental biology*, 82(2):358–370, 1981.

[23] Juan G Cueva, Atticus Mulholland, and Miriam B Goodman. Nanoscale organization of the mec-4 deg/enac sensory mechanotransduction channel in caenorhabditis elegans touch receptor neurons. *Journal of Neuroscience*, 27(51):14089–14098, 2007.

[24] Valeria Vásquez, Michael Krieg, Dean Lockhead, and Miriam B Goodman. Phospholipids that contain polyunsaturated fatty acids enhance neuronal cell mechanics and touch sensation. *Cell reports*, 6(1):70–80, 2014.

[25] Miriam B Goodman and Piali Sengupta. How caenorhabditis elegans senses mechanical stress, temperature, and other physical stimuli. *Genetics*, 212(1):25–51, 2019.

[26] Michael Krieg, Jan Stühmer, Juan G Cueva, Richard Fetter, Kerri Spilker, Daniel Cremers, Kang Shen, Alexander R Dunn, and Miriam B Goodman. Genetic defects in β-spectrin and tau sensitize c. elegans axons to movement-induced damage via torque-tension coupling. *Elife*, 6:e20172, 2017.

[27] Christophe Leterrier, Pankaj Dubey, and Subhojit Roy. The nano-architecture of the axonal cytoskeleton. *Nature Reviews Neuroscience*, 18(12):713, 2017.

[28] Lesley Emtage, Guoqiang Gu, Erika Hartwieg, and Martin Chalfie. Extracellular proteins organize the mechanosensory channel complex in c. elegans touch receptor neurons. *Neuron*, 44(5):795–807, 2004.

[29] Sergei Sukharev and David P Corey. Mechanosensitive channels: multiplicity of families and gating paradigms. *Science's STKE*, 2004(219):re4–re4, 2004.

[30] Benjamin D Matthews, Charles K Thodeti, and Donald E Ingber. Activation of mechanosensitive ion channels by forces transmitted through integrins and the cytoskeleton. *Current Topics in Membranes*, 58:59–85, 2007.

[31] Boris Martinac. The ion channels to cytoskeleton connection as potential mechanism of mechanosensitivity. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1838(2):682–691, 2014.

[32] Charles D Cox, Chilman Bae, Lynn Ziegler, Silas Hartley, Vesna Nikolova-Krstevski, Paul R Rohde, Chai-Ann Ng, Frederick Sachs, Philip A Gottlieb, and Boris Martinac. Removal of the mechanoprotective influence of the cytoskeleton reveals piezo1 is gated by bilayer tension. *Nature communications*, 7(1):1–13, 2016.

[33] Martin Chalfie and J Nichol Thomson. Organization of neuronal microtubules in the nematode caenorhabditis elegans. *The Journal of cell biology*, 82(1):278–289, 1979.

[34] Alessandro Sanzeni, Samata Katta, Bryan Petzold, Beth L Pruitt, Miriam B Goodman, and Massimo Vergassola. Somatosensory neurons integrate the geometry of skin deformation and mechanotransduction channels to shape touch sensing. *Elife*, 8:e43226, 2019.

[35] Michael Krieg, Alexander R Dunn, and Miriam B Goodman. Mechanical control of the sense of touch by β-spectrin. *Nature cell biology*, 16(3):224–233, 2014.

[36] Victor Marc Nigon and Marie-Anne Félix. History of research on c. elegans and other free-living nematodes as model organisms. *WormBook: The Online Review of C. elegans Biology [Internet]*, 2018.

[37] Edward M Hedgecock, JOSEPH G Culotti, David H Hall, and BRIAN D Stern. Genetics of cell and axon migrations in caenorhabditis elegans. *Development*, 100(3):365–382, 1987.

[38] Matthew Buechner, David H Hall, Harshida Bhatt, and Edward M Hedgecock. Cystic canal mutants in caenorhabditis elegans are defective in the apical membrane domain of the renal (excretory) cell. *Developmental biology*, 214(1):227–241, 1999.

[39] Lois G Edgar. Blastomere culture and analysis. *Methods in cell biology*, 48:303–321, 1995.

[40] Laird Bloom. Genetic and molecular analysis of genes required for axon outgrowth in caenorhabditis elegans. 1994.

[41] Michael Christensen, Ana Estevez, Xiaoyan Yin, Rebecca Fox, Rebecca Morrison, Maureen McDonnell, Christina Gleason, David M Miller III, and Kevin Strange. A primary culture system for functional analysis of c. elegans neurons and muscle cells. *Neuron*, 33(4):503–514, 2002.

[42] Yun Zhang, Charles Ma, Thomas Delohery, Brian Nasipak, Barrett C Foat, Alexander Bounoutas, Harmen J Bussemaker, Stuart K Kim, and Martin Chalfie. Identification of genes expressed in c. elegans touch receptor neurons. *Nature*, 418(6895):331–335, 2002.

[43] Marc E Colosimo, Adam Brown, Saikat Mukhopadhyay, Christopher Gabel, Anne E Lanjuin, Aravinthan DT Samuel, and Piali Sengupta. Identification of thermosensory and olfactory neuron-specific genes via expression profiling of single neuron types. *Current biology*, 14(24):2245–2251, 2004.

[44] Hulusi Cinar, Sunduz Keles, and Yishi Jin. Expression profiling of gabaergic motor neurons in caenorhabditis elegans. *Current Biology*, 15(4):340–346, 2005.

[45] Rebecca M Fox, Stephen E Von Stetina, Susan J Barlow, Christian Shaffer, Kellen L Olszewski, Jason H Moore, Denis Dupuy, Marc Vidal, and David M Miller. A gene expression fingerprint of c. elegans embryonic motor neurons. *BMC genomics*, 6(1):1–23, 2005.

[46] Rebecca M Fox, Joseph D Watson, Stephen E Von Stetina, Joan McDermott, Thomas M Brodigan, Tetsunari Fukushige, Michael Krause, and David M Miller. The embryonic muscle transcriptome of caenorhabditis elegans. *Genome biology*, 8(9):1–20, 2007.

[47] Dean Lockhead, Erich M Schwarz, Robert O'Hagan, Sebastian Bellotti, Michael Krieg, Maureen M Barr, Alexander R Dunn, Paul W Sternberg, and Miriam B Goodman. The tubulin repertoire of caenorhabditis elegans sensory neurons and its context-dependent role in process outgrowth. *Molecular Biology of the Cell*, 27(23):3717–3728, 2016.

[48] Amanda R Burnham-Marusich, Casey J Snodgrass, Anna M Johnson, Conrad M Kiyoshi, Sarah E Buzby, Matt R Gruner, and Patricia M Berninsone. Metabolic labeling of caenorhabditis elegans primary embryonic cells with azido-sugars as a tool for glycoprotein discovery. *PloS one*, 7(11):e49020, 2012.

[49] Raja Settivari, Jennifer LeVora, and Richard Nass. The divalent metal transporter homologues smf-1/2 mediate dopamine neuron sensitivity in caenorhabditis elegans models of manganism and parkinson disease. *Journal of Biological Chemistry*, 284(51):35758–35768, 2009.

[50] Laura Bianchi, Beate Gerstbrein, Christian Frøkjær-Jensen, Dewey C Royal, Gargi Mukherjee, Mary Anne Royal, Jian Xue, William R Schafer, and Monica Driscoll. The neurotoxic mec-4 (d) deg/enac sodium channel conducts calcium: implications for necrosis initiation. *Nature neuroscience*, 7(12):1337–1344, 2004.

[51] Dieter R Klopfenstein and Ronald D Vale. The lipid binding pleckstrin homology domain in unc-104 kinesin is necessary for synaptic vesicle transport in caenorhabditis elegans. *Molecular biology of the cell*, 15(8):3729–3739, 2004.

[52] H Robert Horvitz, Sydney Brenner, Jonathan Hodgkin, and Robert K Herman. A uniform

genetic nomenclature for the nematode caenorhabditis elegans. *Molecular and General Genetics MGG*, 175(2):129–133, 1979.

[53] Amy L Eastwood and Miriam B Goodman. Insight into deg/enac channel gating from genetics and structure. *Physiology*, 27(5):282–290, 2012.

[54] Sylvia Fechner, Isabel D'Alessandro, Lingxin Wang, Calvin Tower, Li Tao, and Miriam B Goodman. Deg/enac/asic channels vary in their sensitivity to anti-hypertensive and non-steroidal anti-inflammatory drugs. *Journal of General Physiology*, 153(4), 2021.

[55] Kyonsoo Hong, Itzhak Mano, and Monica Driscoll. In vivo structure–function analyses of-caenorhabditis elegans mec-4, a candidate mechanosensory ion channel subunit. *Journal of Neuroscience*, 20(7):2575–2588, 2000.

[56] Monica Driscoll and Martin Chalfie. The mec-4 gene is a member of a family of caenorhabditis elegans genes that can mutate to induce neuronal degeneration. *Nature*, 349(6310):588–593, 1991.

[57] Hiroshi Suzuki, Rex Kerr, Laura Bianchi, Christian Frøkjær-Jensen, Dan Slone, Jian Xue, Beate Gerstbrein, Monica Driscoll, and William R Schafer. In vivo imaging of c. elegans mechanosensory neurons demonstrates a specific role for the mec-4 channel in the process of gentle touch sensation. *Neuron*, 39(6):1005–1017, 2003.

[58] Miriam B Goodman, David H Hall, Leon Avery, and Shawn R Lockery. Active currents regulate sensitivity and dynamic range in c. elegans neurons. *Neuron*, 20(4):763–772, 1998.

[59] Jóhanna Árnadóttir, Robert O'Hagan, Yushu Chen, Miriam B Goodman, and Martin Chalfie. The deg/enac protein mec-10 regulates the transduction channel complex in caenorhabditis elegans touch receptor neurons. *Journal of Neuroscience*, 31(35):12695–12704, 2011.

[60] Xiaoyin Chen and Martin Chalfie. Regulation of mechanosensation in c. elegans through ubiquitination of the mec-4 mechanotransduction channel. *Journal of Neuroscience*, 35(5):2200–2212, 2015.

[61] Samata Katta, Alessandro Sanzeni, Alakananda Das, Massimo Vergassola, and Miriam B Goodman. Progressive recruitment of distal mec-4 channels determines touch response strength in c. elegans. *Journal of General Physiology*, 151(10):1213–1230, 2019.

[62] Boris Martinac and CD Cox. Mechanosensory transduction: focus on ion channels. *Reference Module in Life Sciences; Elsevier BV: Amsterdam, The Netherlands*, 2017.

[63] Andriy Anishkin, Stephen H Loukin, Jinfeng Teng, and Ching Kung. Feeling the hidden mechanical forces in lipid bilayer is an original sense. *Proceedings of the National Academy of Sciences*, 111(22):7898–7905, 2014.

[64] Julio F Cordero-Morales and Valeria Vásquez. How lipids contribute to ion channel function, a fat perspective on direct and indirect interactions. *Current opinion in structural biology*, 51:92–98, 2018.

[65] Eduardo Perozo, Anna Kloda, D Marien Cortes, and Boris Martinac. Physical principles underlying the transduction of bilayer deformation forces during mechanosensitive channel gating. *Nature structural biology*, 9(9):696–703, 2002.

[66] Erdinc Sezgin, Ilya Levental, Satyajit Mayor, and Christian Eggeling. The mystery of membrane organization: composition, regulation and roles of lipid rafts. *Nature reviews Molecular cell biology*, 18(6):361, 2017.

[67] Clare M O'Connor, Jill U Adams, and Jennifer Fairman. Essentials of cell biology. *Cambridge, MA: NPG Education*, 1:54, 2010.

[68] Pietro Ridone, Stephan L Grage, Amrutha Patkunarajah, Andrew R Battle, Anne S Ulrich, and Boris Martinac. "force-from-lipids" gating of mechanosensitive channels modulated by pufas. *Journal of the mechanical behavior of biomedical materials*, 79:158–167, 2018.

[69] Shifang Zhang, Johanna Arnadottir, Charles Keller, Guy A Caldwell, C Andrea Yao, and Martin Chalfie. Mec-2 is recruited to the putative mechanosensory complex in c. elegans touch receptor neurons through its stomatin-like domain. *Current Biology*, 14(21):1888–1896, 2004.

[70] Miriam B Goodman, Glen G Ernstrom, Dattananda S Chelur, Robert O'Hagan, C Andrea Yao, and Martin Chalfie. Mec-2 regulates c. elegans deg/enac channels needed for mechanosensation. *Nature*, 415(6875):1039–1042, 2002.

[71] Austin L Brown, Zhiwen Liao, and Miriam B Goodman. Mec-2 and mec-6 in the caenorhabditis elegans sensory mechanotransduction complex: auxiliary subunits that enable channel activity. *The Journal of general physiology*, 131(6):605–616, 2008.

[72] Tobias B Huber, Bernhard Schermer, Roman Ulrich Müller, Martin Höhne, Malte Bartram, Andrea Calixto, Henning Hagmann, Christian Reinhardt, Fabienne Koos, Karl Kunzelmann, et al. Podocin and mec-2 bind cholesterol to regulate the activity of associated ion channels. *Proceedings of the National Academy of Sciences*, 103(46):17079–17086, 2006.

[73] Adam L Nekimken, Beth L Pruitt, and Miriam B Goodman. Touch-induced mechanical strain in somatosensory neurons is independent of extracellular matrix mutations in caenorhabditis elegans. *Molecular biology of the cell*, 31(16):1735–1743, 2020.

[74] Hongping Du, Guoqiang Gu, Chris M William, and Martin Chalfie. Extracellular proteins needed for c. elegans mechanosensation. *Neuron*, 16(1):183–194, 1996.

[75] Junyue Cao, Jonathan S Packer, Vijay Ramani, Darren A Cusanovich, Chau Huynh, Riza Daza, Xiaojie Qiu, Choli Lee, Scott N Furlan, Frank J Steemers, et al. Comprehensive single-cell transcriptional profiling of a multicellular organism. *Science*, 357(6352):661–667, 2017.

[76] Rachele Sangaletti, Gerhard Dahl, and Laura Bianchi. Mechanosensitive unpaired innexin channels in c. elegans touch neurons. *American Journal of Physiology-Cell Physiology*, 307(10):C966–C977, 2014.

[77] Denise S Walker and William R Schafer. Distinct roles for innexin gap junctions and hemichannels in mechanosensation. *Elife*, 9:e50597, 2020.

[78] Martin Chalfie, Anne C Hart, Catharine H Rankin, and Miriam B Goodman. Assaying mechanosensation. *WormBook: the online review of C. elegans biology*, 2014.

[79] S-J Park, B Petzold, MB Goodman, and BL Pruitt. Piezoresistive cantilever-based force-clamp system for the study of mechanotransduction in c. elegans. In *2009 IEEE 22nd International Conference on Micro Electro Mechanical Systems*, pages 188–191. IEEE, 2009.

[80] Amy L Eastwood, Alessandro Sanzeni, Bryan C Petzold, Sung-Jin Park, Massimo Vergassola, Beth L Pruitt, and Miriam B Goodman. Tissue mechanics govern the rapidly adapting and symmetrical response to touch. *Proceedings of the National Academy of Sciences*, 112(50):E6955–E6963, 2015.

[81] Adam L Nekimken, Holger Fehlauer, Anna A Kim, Sandra N Manosalvas-Kjono, Purim Ladpli, Farah Memon, Divya Gopisetty, Veronica Sanchez, Miriam B Goodman, Beth L Pruitt, et al. Pneumatic stimulation of c. elegans mechanoreceptor neurons in a microfluidic trap. *Lab on a Chip*, 17(6):1116–1127, 2017.

[82] Elisabeth A Cox and Jeff Hardin. Sticky worms: adhesion complexes in c. elegans. *Journal of cell science*, 117(10):1885–1897, 2004.

[83] Yushu Chen, Shashank Bharill, Zeynep Altun, Robert O'Hagan, Brian Coblitz, Ehud Y Isacoff, and Martin Chalfie. Caenorhabditis elegans paraoxonase-like proteins control the functional expression of deg/enac mechanosensory proteins. *Molecular biology of the cell*, 27(8):1272–1285, 2016.

[84] Eileen A Mazzochette. *An integrated and automated system for assaying touch sensation in freely moving C. elegans*. Stanford University, 2016.

[85] Harper C VanSteenhouse, Zachary A Horton, Robert O'Hagan, Mei-Hui Tai, and Birgit Zipser. Phylogenetic conservation of the cell-type-specific lan3-2 glycoepitope in caenorhabditis elegans. *Development genes and evolution*, 220(3-4):77–87, 2010.

[86] Michelle L Tucci, Guy A Caldwell, and Kim A Caldwell. Cell culture to investigate neurotoxicity and neurodegeneration utilizing caenorhabditis elegans. In *Cell Culture Techniques*, pages 129–143. Springer, 2011.

[87] Raja Settivari, Natalia VanDuyn, Jennifer LeVora, and Richard Nass. The nrf2/skn-1-dependent glutathione s-transferase π homologue gst-1 inhibits dopamine neuron degeneration in a caenorhabditis elegans model of manganism. *Neurotoxicology*, 38:51–60, 2013.

[88] Kevin Strange, Michael Christensen, and Rebecca Morrison. Primary culture of caenorhabditis elegans developing embryo cells for electrophysiological, cell biological and molecular studies. *Nature protocols*, 2(4):1003, 2007.

[89] Rachele Sangaletti and Laura Bianchi. A method for culturing embryonic c. elegans cells. *Journal of visualized experiments: JoVE*, (79), 2013.

[90] Qun Zheng, Shikha Ahlawat, Anneliese Schaefer, Tim Mahoney, Sandhya P Koushika, and Michael L Nonet. The vesicle protein sam-4 regulates the processivity of synaptic vesicle transport. *PLoS Genet*, 10(10):e1004644, 2014.

[91] Yinhua Zhang, Jeremy M Foster, Laura S Nelson, Dong Ma, and Clotilde KS Carlow. The chitin synthase genes chs-1 and chs-2 are essential for c. elegans development and responsible for chitin deposition in the eggshell and pharynx, respectively. *Developmental biology*, 285(2):330–339, 2005.

[92] Dean Lockhead. *A Study of Environmental Contributions to the Growth and Function of the Gentle Touch Receptor Neurons in Caenorhabditis elegans*. Stanford University, 2016.

[93] Zeynep Altun-Gultekin, Yoshiki Andachi, Ephraim L Tsalik, David Pilgrim, Yuji Kohara, and Oliver Hobert. A regulatory cascade of three homeobox genes, ceh-10, ttx-3 and ceh-23, controls cell fate specification of a defined interneuron class in c. elegans. *Development*, 128(11):1951–1969, 2001.

[94] Florian Hahne, Nolwenn LeMeur, Ryan R Brinkman, Byron Ellis, Perry Haaland, Deepayan Sarkar, Josef Spidlen, Errol Strain, and Robert Gentleman. flowcore: a bioconductor package for high throughput flow cytometry. *BMC bioinformatics*, 10(1):1–8, 2009.

[95] Phu Van, Wenxin Jiang, Raphael Gottardo, and Greg Finak. ggcyto: next generation open-source visualization software for cytometry. *Bioinformatics*, 34(22):3951–3953, 2018.

[96] Mario Roederer. Compensation in flow cytometry. *Current protocols in cytometry*, 22(1):1–14, 2002.

[97] David J Klinke and Kathleen M Brundage. Scalable analysis of flow cytometry data using r/bioconductor. *Cytometry Part A: The Journal of the International Society for Advancement of Cytometry*, 75(8):699–706, 2009.

[98] Kieran O'Neill, Nima Aghaeepour, Josef Špidlen, and Ryan Brinkman. Flow cytometry bioinformatics. *PLoS Comput Biol*, 9(12):e1003365, 2013.

[99] Anja B Bohn, Bjarne K Moller, and Mikkel S Petersen. Flow cytometry and compensation of highly autofluorescent cells: the example of mesenchymal stem cells. *Stem Cell Biology and Research*, 2(1):4, 2015.

[100] Stephen P Perfetto, Pratip K Chattopadhyay, Laurie Lamoreaux, Richard Nguyen, David Ambrozak, Richard A Koup, and Mario Roederer. Amine-reactive dyes for dead cell discrimination in fixed samples. *Current protocols in cytometry*, 53(1):9–34, 2010.

[101] Mario Roederer and Robert F Murphy. Cell-by-cell autofluorescence correction for low signal-to-noise systems: Application to epidermal growth factor endocytosis by 3t3 fibroblasts. *Cytometry: The Journal of the International Society for Analytical Cytology*, 7(6):558–565, 1986.

[102] Jennifer I Semple, Rosa Garcia-Verdugo, and Ben Lehner. Rapid selection of transgenic c. elegans using antibiotic resistance. *Nature methods*, 7(9):725–727, 2010.

[103] Daniel J Dickinson, Ariel M Pani, Jennifer K Heppert, Christopher D Higgins, and Bob Goldstein. Streamlined genome engineering with a self-excising drug selection cassette. *Genetics*, 200(4):1035–1049, 2015.

[104] Johanna Michl, Kyung Chan Park, and Pawel Swietach. Evidence-based guidelines for controlling ph in mammalian live-cell culture systems. *Communications biology*, 2(1):1–12, 2019.

[105] Daniel G Gibson, Lei Young, Ray-Yuan Chuang, J Craig Venter, Clyde A Hutchison, and Hamilton O Smith. Enzymatic assembly of dna molecules up to several hundred kilobases. *Nature methods*, 6(5):343–345, 2009.

[106] Didier Falconnet, Gabor Csucs, H Michelle Grandin, and Marcus Textor. Surface engineering approaches to micropattern surfaces for cell-based assays. *Biomaterials*, 27(16):3044–3063, 2006.

[107] Jens Moeller, Aleksandra K Denisin, Joo Yong Sim, Robin E Wilson, Alexandre JS Ribeiro, and Beth L Pruitt. Controlling cell shape on hydrogels using lift-off protein patterning. *PloS one*, 13(1):e0189901, 2018.

[108] Pierre-Olivier Strale, Ammar Azioune, Ghislain Bugnicourt, Yohan Lecomte, Makhlad Chahid, and Vincent Studer. Multiprotein printing by light-induced molecular adsorption. *Advanced Materials*, 28(10):2024–2029, 2016.

[109] Leeya Engel, Guido Gaietta, Liam P Dow, Mark F Swif, Gaspard Pardon, Niels Volkmann, William I Weis, Dorit Hanein, and Beth L Pruitt. Extracellular matrix micropatterning technology for whole cell cryogenic electron microscopy studies. *Journal of Micromechanics and Microengineering*, 29(11):115018, 2019.

[110] Wayne S Rasband. National institutes of health, bethesda, maryland, usa. *http://imagej. nih. gov/ij/*, 2011.

[111] Alexandre JS Ribeiro, Kathia Zaleta-Rivera, Euan A Ashley, and Beth L Pruitt. Stable, covalent attachment of laminin to microposts improves the contractility of mouse neonatal cardiomyocytes. *ACS applied materials & interfaces*, 6(17):15516–15526, 2014.

[112] Daniel P Keeley, Eric Hastie, Ranjay Jayadev, Laura C Kelley, Qiuyi Chi, Sara G Payne, Jonathan L Jeger, Brenton D Hoffman, and David R Sherwood. Comprehensive endogenous tagging of basement membrane components reveals dynamic movement within the matrix scaffolding. *Developmental cell*, 54(1):60–74, 2020.

[113] Jeremy M Berg, John L Tymoczko, and Lubert Stryer. Lectins are specific carbohydrate-binding proteins. *Biochemistry*, pages 333–335, 2002.

[114] Bryan Carl Petzold. *The Role of Body Mechanics in Touch Sensation in the Nematode Caenorhabditis elegans*. Stanford University, 2013.

[115] Yushu Chen, Shashank Bharill, Robert O'Hagan, Ehud Y Isacoff, and Martin Chalfie. Mec-10 and mec-19 reduce the neurotoxicity of the mec-4 (d) deg/enac channel in caenorhabditis elegans. *G3: Genes, Genomes, Genetics*, 6(4):1121–1130, 2016.