Spring 2021

Teaching Computation in Neuroscience: Notes on the 2019 Society for Neuroscience Professional Development Workshop on Teaching

William Grisham
University of California, Los Angeles

Mathew Abrams
Karolinska Institute

Walt E. Babiec
University of California, Los Angeles

Adriene L. Fairhall
University of Washington

Robert E. Kass
Carnegie Mellon University

See next page for additional authors

ARTICLE
Teaching Computation in Neuroscience: Notes on the 2019 Society for Neuroscience Professional Development Workshop on Teaching

William Grisham[1], Mathew Abrams[2], Walt E. Babiec[3], Adrienne L. Fairhall[4], Robert E. Kass[5], Pascal Wallisch[6], and Richard Olivo[7]

[1]Department of Psychology, UCLA, Los Angeles, CA, 90095-1563; [2]International Neuroinformatics Coordinating Facility, Karolinska Institutet. Nobels väg 15A, Stockholm. Sweden SE-171 77; [3]Neuroscience Interdepartmental Program / Physiology, UCLA, Los Angeles, CA, 90095-1761; [4]Department of Physiology and Biophysics and Computational Neuroscience Center, University of Washington, Seattle WA 98195; [5] Department of Statistics & Data Science, Machine Learning Department, and Neuroscience Institute, Carnegie Mellon University, Pittsburgh, PA 15213; [6]Department of Psychology, New York University, New York, NY 10003; [7]Department of Biological Sciences, Smith College, Northampton, MA 01063.

The 2019 Society for Neuroscience Professional Development Workshop on Teaching reviewed current tools, approaches, and examples for teaching computation in neuroscience. Robert Kass described the statistical foundations that students need to properly analyze data. Pascal Wallisch compared MATLAB and Python as programming languages for teaching students. Adrienne Fairhall discussed computational methods, training opportunities, and curricular considerations. Walt Babiec provided a view from the trenches on practical aspects of teaching computational neuroscience. Mathew Abrams concluded the session with an overview of resources for teaching and learning computational modeling in neuroscience.

Key words: Society for Neuroscience; teaching workshop; professional development; computational neuroscience; coding, programming, MATLAB, Python, modeling

---

If the human brain were so simple
That we could understand it,
We would be so simple
That we couldn't.
- George Edgin Pugh, *The Biological Origin of Human Values*, 1977

The task of understanding brains is a central aim of neuroscience. As educators, we need ways of conceptualizing the brain so that we can explain it and its function to students. Models can fit this need if they reflect important aspects of reality. A plastic model of a human brain reflects reality and can explain neuroanatomy, but it is static. Real brains, by contrast, are complex, dynamic, and interactive -- often in a nonlinear fashion across time. Thus, if we are to capture this reality, we need effective models, and the only models that could reasonably fulfill this role are computational ones. In addition, stunning advances in recording, molecular, and anatomical techniques provide us with data sets of ever-increasing complexity, pushing the need for tools and concepts to extract meaning from these data. The BRAIN Initiative's BRAIN 2025 report (Bargmann et al., 2014) put theory, modeling and data analysis at the core of expected future advances in neuroscience, and the 2019 BRAIN review (du Lac et al, 2019) underscored the ongoing pressing need for training in these areas.

Teaching computational neuroscience endows students with valuable skills as they enter the workforce (Grisham et al., 2016). Indeed, the National Science Foundation (NSF) and American Association for the Advancement of Science Vision and Change (AAAS, 2011) document urges educators in biological science to augment students' quantitative reasoning by using modeling and simulation to describe living systems. Statistical methods of analyzing data are best learned in pursuit of scientific questions, and such experience and skills in data science have never been in greater demand. At one time this curriculum seemed beyond the reach of most undergraduates (Grisham, 2014), but a reconsideration of the zeitgeist forces us to conclude that now is the time to develop such courses. The Professional Development Workshop on Teaching at the 2019 annual meeting of the Society for Neuroscience gathered together experts to discuss options for teaching computation in neuroscience, with the goal of helping faculty plan or revise courses in this area, particularly for undergraduates.

ROB KASS: STATISTICAL BACKGROUND AND STATISTICAL MODELS IN COMPUTATIONAL NEUROSCIENCE: WHAT IS COMPUTATIONAL NEUROSCIENCE?

Computational neuroscience emerged from converging ideas that would now be associated with computer science, mathematics, neuroscience, psychology, and statistics. It remains helpful for students, even within the briefest of introductions, to appreciate the very constructive interplay among multiple disciplines in attempting to understand the brain. With support of an NIH Blueprint training grant, for the past three years the initial pages from Kass et al. (2018)

have served as an introductory reading in many contexts, including an undergraduate bootcamp in computational neuroscience for students from across the country.

There are two distinct ways that statistics enter computational neuroscience: first, through stochastic models of neural phenomena, and second, through data analysis. Students should have a feeling for both of these roles of statistics. It would be possible to design an undergraduate curriculum on the basics of computational neuroscience that brings in these two roles of statistics. An introductory course in this curriculum could serve both computational and non-computational students. For many years, Robert Kass has taught such an introductory course in computational neuroscience to graduate students from a range of programs — from biology to engineering — at the Center for the Neural Basis of Cognition (a joint effort of Carnegie Mellon and the University of Pittsburgh). Two projects are currently underway to provide educators with new resources: a textbook on computational neuroscience, and a collection of 10-minute videos on selected topics. It will likely be several years until the textbook is available, but the videos should be public by 2021 (and the collection is designed to grow with contributions from instructors and researchers around the world).

The constraints of most undergraduate curricula, however, are often limited by scope and faculty expertise, and it isn't clear how many institutions will soon be able to accommodate a semester-long course on computational neuroscience for undergraduates. Furthermore, a designer of such a course faces a choice: either accept some superficiality and teach to diverse backgrounds, or require multiple prerequisites in math and statistics as well as programming comfort with high-level languages such as Python, MATLAB, or R. Many neuroscience instructors might like to incorporate some computational topics into a general neuroscience course, but even at this level, background lectures are essential, and material from Kass et al. (2014) will be helpful because it is aimed at an undergraduate neuroscience audience.

Ten essential topics in computational neuroscience, including four on background material, would be:

1. Random variables and important probability distributions.
2. Random vectors, least-squares linear regression, and the underlying linear algebra.
3. Bayes' Theorem and the optimality of Bayes classifiers; the Law of Large Numbers and the Central Limit Theorem; and statistical estimation.
4. The exponential function solutions to a first-order differential equation.
5. Random walk models of integrate-and-fire functions of neurons; effects of noise: balanced excitation and inhibition.
6. Electrical circuit model of a neuron. Passive synaptic dynamics and phenomenological models of spiking and integrate-and-fire dynamics.
7. The Hodgkin-Huxley model of action potential generation.
8. Population vectors.
9. Information theory in human discrimination. A nice reading is Miller (1956).
10. Cognition and optimality. An overview is given in Chapter 1 of Anderson (2007).

PASCAL WALLISCH: TEACHING A PROGRAMMING LANGUAGE: MATLAB OR PYTHON?

Computational neuroscience analyses of data can be broken down into three levels At the top, data sets usually are multivariate, so the **strategic goal** is often to reduce dimensions, which is the task of an algorithm. **Algorithms**, the second level, are tactics to achieve a strategic goal, and there are many algorithms to choose from. After choosing one, there is an **implementation stage** where coding takes place, the third level. Students usually focus on the implementation level, which is one level down from the algorithmic level. Nonetheless, as educators, we should urge students to focus on the algorithmic level and ask questions such as, "Does the algorithm fit the problem?" and "Do the data conform to assumptions of the algorithmic tactic?" Answers to these questions should determine the choice of the algorithmic tactic—and hence the programming package.

Programs are lifeboats to keep students from drowning in the tsunami of data. The two lifeboats currently receiving attention are MATLAB & Python. MATLAB stands for "matrix laboratory" and was actually created to teach FORTRAN. Python's name comes from Monty Python, not from the snake. Both are high level programming languages.

Although social media discussions about the two languages are sometimes quite vehement, there really is no need to be dogmatic in choosing between them. Both are tools that allow one to achieve some goal from an initial state. A tool removes a problem standing in the way of achieving a goal. So, what's the best tool? That depends where you start, what you want to do, and what the problem is. The tool should fit the problem, and it is okay to use more than one tool.

There are various considerations for choosing between MATLAB or Python. One is processing speed, which isn't very different between the two because both are fairly fast. Another is cognitive ease, which is a measure of the difficulty of writing and understanding code. The two are fairly comparable — both MATLAB and Python are high level programs with thousands of functions available. Another consideration is backward compatibility — MathWorks makes sure that backward compatibility exists for prior code, and although Python is more leading edge, its developers don't seem terribly concerned with backward compatibility. One of the biggest conceptual differences between the two is indexing values: Python starts at 0, MATLAB starts at 1. Also, the data type is a matrix in MATLAB, whereas Python is general purpose. Which you choose to use will depend on the task you want to accomplish.

Style is also different between the two packages; blank space is meaningful in Python but not in MATLAB. Despite

arguments on social media sites, Python is not really more elegant than MATLAB. MATLAB has a more straightforward syntax, and Python is often more verbose. Both packages are continually evolving to improve their capabilities and ease of use. Python is now easier to install than it was before Anaconda was developed. New MATLAB capabilities of string handling datatypes are now expanded. Webscraping is easier in Python, but sound-handling is easier in MATLAB. Again, the choice depends on the problem.

Although Python is currently the most popular language among programmers, many more publications have used MATLAB rather than Python for data analyses; Python and its variants make up less than 1% of publications.

The most compelling issue in making a choice is where your students are in their programming abilities. Inexperienced students in NYU classes do better with MATLAB, and MATLAB provides excellent support with actual people available to help 24/7. With Python, there is no one to call, and Stack Overflow is the help desk. The problem is that Stack Overflow is a wiki that may or may not have the right information.

Finally, there are cost considerations—MATLAB is easier to learn, but requires a license whereas Python is free. Dr. Wallisch's book with Eric Nylan (2017), *Neural Data Science: A Primer with MATLAB and Python*, teaches both in Rosetta Stone fashion by providing programs in both languages along with English explanations.

So to answer the question, "Which one should an educator pick?" Whatever works for you and your students.

ADRIENNE FAIRHALL: TEACHING COMPUTATIONAL NEUROSCIENCE

Computational neuroscience as a field is the union of multiple approaches to understanding neural function: theory, modeling, and data analysis (Figure 1). *Theory* is the big picture: the algorithm or framework or principle or solution space that underlies a specific dynamic. Examples include reinforcement learning (algorithm), Hopfield networks (framework), efficient coding (principle) and attractor dynamics (solution space). *Modeling* describes the attempt to write down and solve equations that reproduce aspects of experimental data, however coarse-grained. A classic example is the Hodgkin-Huxley system of equations for action potential generation in an axon, or at the opposite extreme, the Blue Brain project, which aims to simulate at biophysical levels of detail the activity of an entire cortical column (Einevoll et al., 2019). The goal of *data analysis* is to characterize a system via observations (Aljadeff et al., 2016; Kass et al., 2014), ideally revealing properties or dynamics that can be mapped onto models and, ultimately, test theories.

Teaching computational neuroscience is, therefore, challenging and multifaceted. Full understanding of a neural system should involve all three components (Fairhall, 2014), so students should gain some facility with all of these aspects. Each aspect involves distinct disciplines of mathematics, engineering, computer science, physics, and statistics.

Challenges

Students have different needs or expectations for their training. All emerging systems neuroscientists should gain sufficient mathematical background and coding skill so they can manipulate data. While some will want to attain proficiency in order to understand and use ideas and methods in experimental research, others want to specialize in theory. However, to be able to develop novel conceptual theories or devise new data analysis methods, it is certainly helpful to have a deep grasp of at least one quantitative field — applied mathematics, statistics, physics, or computer science. Thus, designing a training program in computational neuroscience at both undergraduate and graduate levels needs to handle diverse preparation and expectations. Further, a program needs to provide a breadth of understanding and a grounding in basic neuroscience with the opportunity for depth of training in a specific field. Given that students enter with a wide range of backgrounds, students and teachers can help to bridge some of the inevitable gaps.

Core Coursework

An ideal minimal coursework sequence will likely need to bridge undergraduate and graduate classes. Given the increasing sophistication of analysis required for large data sets, undergraduate students thinking of entering systems neuroscience need to maintain a reasonably high level of core mathematics, whether they plan to become theorists or experimentalists. Providing early information and encouragement to new undergraduates to maintain mathematical training can have high impact. For example, early exposure in the freshman or sophomore year to neuroscience research talks can highlight the deep intersections of neuroscience and mathematical topics, and encourage students to maintain a high level of quantitative undergraduate coursework.

A sequence such as in Table 1 is recommended. Learning to code is obviously vital. This learning could be done in a computer science class but can also be learned

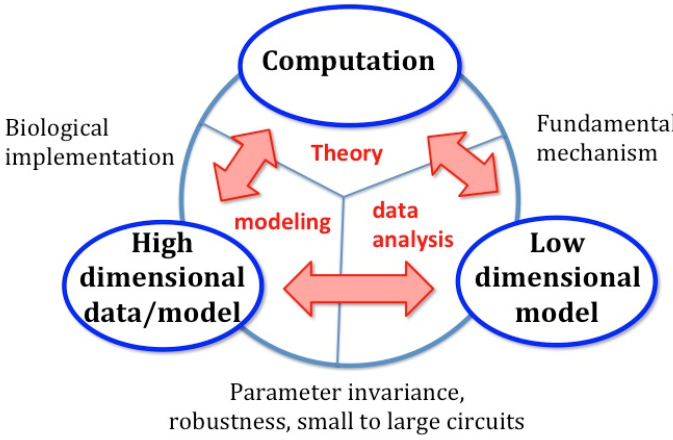

Figure 1. Schema of conceptual areas within computational neuroscience and their roles in understanding biological computation. (Adapted from Fairhall, 2014).

Course work in a computational neuroscience sequence	
Core background • Linear algebra • Differential equations • Probability and statistics	
Theory and modeling • Dynamical systems • Nonlinear dynamics • Statistical physics • Control theory	Data analysis • Statistics • Signal processing • Machine learning
Integrative • Computational neuroscience • Neural network theory • Reinforcement learning • Applications (motor control, sensory systems) • Journal club	

Table 1. Topics for course work.

alongside mathematics or data analysis methods or in the context of an "integrative" computational neuroscience class.

Bridging classes

Graduate school is not too late to learn computational methods. Many institutions are now offering graduate classes in mathematical/quantitative methods for neuroscience. The University of Washington teaches a class called "Quantitative Methods in Neuroscience," offered both for computational neuroscience undergraduates and as a core class for all neuroscience graduate students. The course consists of five modules: linear algebra, differential equations, Fourier transforms, stochastic processes, and principal component analysis. Each topic is studied for two weeks: one week of lectures and one week of interactive exercises buttressed by a classic neuroscience paper that students present that employs the method in the exercises. Bringing together relatively mathematically sophisticated undergraduates and potentially mathematically naïve graduate students allows active two-way exchange in class exercises and presentations. A key tool for this class is a set of MATLAB tutorials, which intersperse liberally commented code with pedagogical text and quiz prompts. In these tutorials students:

- Walk through basic commands with annotations
- Perform computations and parameter variations
- Describe outputs
- Interpret results
- Are prompted to write new code
- Ponder/answer embedded open-ended questions
- Can repurpose code in a novel way in a project.

The outcome for this class is that all students gain a working ability to use MATLAB, and they gain exposure to mathematical ideas in sufficient depth to inspire them to do additional classes or reading where desired. It helps students new to biology to see how mathematics can play an important role in framing and solving biology questions. The compressed format also allows pointing out the relationships between the topics, which are very often obscured when these subjects are learned in isolation — in particular, the key role of linear systems.

Integrative Course Design

In view of the multilevel interactions outlined in Figure 1, it would be ideal when possible to incorporate all three into teaching. For students with diverse preparation, it can be especially important to motivate methods and analysis with a framing of the question being addressed. One can begin with the question posed by the biological system; discuss and explain the big-picture framework and the mathematical underpinnings; consider concrete models; and teach methods to validate or explore models based on data. Incorporating a project for final assessment is an opportunity to consolidate learning of the process of interdisciplinary science.

Be Hands-On

In any computational neuroscience class, students should be coding up models and playing with data. A key teaching decision is: Python or MATLAB? Both have pros and cons, as described in detail by Dr. Wallisch (see above). MATLAB has a lower barrier for entry for newcomers to coding and can be preferable for undergraduates or introductory computational graduate classes that include students who will follow an experimental track. For students specializing in computational fields, Python is a good long-term investment.

Bridging Gaps

Graduate students can pick up missing math as electives or by auditing; postdocs can also audit classes. Online classes between undergraduate and graduate school can be a great way to supplement missing math classes from the "core" list; there are many high-quality options including Khan Academy for basics like linear algebra.

Summer Schools

Summer schools are an excellent accelerated option to gain rapid experience in computational neuroscience. A large number are offered around the US and internationally, helpfully collected by Tom Burns at https://tfburns.github.io/compneuro-summer-schools/ and https://docs.google.com/spreadsheets/d/1b05MPR7bkxnwKjzY-6KHd_aDaV_68qwMhuFVu5IGG9g/edit#gid=276255682. These schools are often suited for more advanced students (senior graduate students and postdocs) but many specifically aim to cater to a wide range of backgrounds and to give a rapid leg up to students who want an intensive learning experience. There are of course many other

benefits to summer schools:

- Students form relationships with international peers which can continue throughout their careers
- Many courses incorporate a project that is an excellent learning opportunity and a chance to work directly with a professor or teaching assistants
- There are opportunities to interact extensively with well-known professors nationally and internationally to build career visibility.

Two highly recommended courses are the Methods in Computational Neuroscience course at the Marine Biological Laboratory and the Summer Workshop for the Dynamic Brain, co-run by the Allen Institute for Brain Science and the University of Washington. Both of these courses mix lectures on systems in neuroscience with mathematics and statistical methods and show how models are developed in application to the systems under discussion. The recent success of Neuromatch Academy (Juavinett, 2020), which offered in depth training to almost 2000 students worldwide online during the 2020 pandemic, is surely going to remain an important model for the future.

Online Classes

Particularly in the wake of COVID-19, online classes provide an important component of computational neuroscience teaching. An online course on the Coursera platform, Computational Neuroscience, has served many students as an introduction to the field (https://www.coursera.org/learn/computational-neuroscience). Coursera also facilitates interaction between students.

WALT BABIEC: GEOMETRY OF THE NERVOUS SYSTEM: A COURSE IN DYNAMICAL SYSTEMS ANALYSIS & MODELING OF NEURAL FUNCTION

Living things change with time. But how do we understand and make sense of that change? Are there no similarities in how organisms change with time? Is there a vocabulary that we can use to describe and differentiate that change? There is, and a UCLA course – Dynamical Systems Modeling of Physiological Systems – provides students in Neuroscience, Physiological Science, and Life Sciences with a rigorous, quantitative framework for describing dynamic behavior in living systems.

The course takes a dynamical systems approach to describing or modeling living systems. The state of any system, whether it be a gene network, a cell, a whole organism, or an ecosystem, can be described by a set of time-varying state variables such as protein concentration, animal population, or genotype prevalence. Differential equations or iterated maps are used to define how those state variables are changing at any instant. They provide the road map to understanding how a system changes with time, whether certain changes with time are even possible, and what might happen to those forms of change if properties of the system change.

Before describing how to teach the dynamical systems approach to students whose first love is not mathematics, it's important to understand the big ideas, the purpose for teaching them this material in the first place. First, it is important to modernize students' view of the meaning of homeostasis. Rather than Cannon's (1929) more rigid notion of keeping things standing still at a certain physiological set point, the course helps students recognize and understand that most homeostatic processes, such as regulation of Purkinje neuron firing rate, hypothalamic control of body temperature, or the sleep-wake and circadian cycles, are controlled oscillations rather than static equilibria. While this may sound a little bit undefined in language, our dynamical systems viewpoint allows us to generate precise definitions that can be tested. If the change in temperature, that is, $T' = 0$, then we have true homeostasis. If, however, $T' \neq 0$, but the trajectory of T repeats with time, we have a stable limit-cycle oscillation. The latter is what we see throughout physiological systems and nature as a whole.

This brings us to the next major point. Real physiological systems are nonlinear, rely upon feedback, and operate with time delays. While we often eschew nonlinearities in engineered systems, they are essential in physiological systems. Simple but incredibly important decisions (for example, whether a neuron fires an action potential or remains quiescent) are nonlinear by their very nature and cannot be linearized without losing that behavior. Furthermore, oscillatory behavior requires negative feedback and time delays in order to operate properly. Again, a dynamical systems approach allows observing and analyzing how the strength of the feedback and the length of the time delays affect the behavior of the physiological system being studied.

Finally, a dynamical systems viewpoint allows us to understand and observe how complex physiological behavior emerges from self-organized activity. For example, every neuroscience student is taught that a neuron has a firing threshold. If the membrane potential stays below threshold, the neuron doesn't fire. If the membrane potential exceeds threshold, the neuron fires an action potential. But where is threshold stored? What protein or nucleotide sequence encodes threshold? After studying the Hodgkin-Huxley equations and Fitzhugh's simplified formulation of them, the students understand that threshold is an emergent property of the nonlinear interaction of the passive membrane properties of the neuron with voltage-activated Na+ and K+ channels. Crossing threshold represents a Hopf bifurcation from equilibrium behavior at rest, to oscillatory behavior during action potential firing. Action potentials, therefore, are an emergent property of solubilized ions, lipids, and proteins whose dynamic interactions can be described effectively and efficiently with a system of four ordinary differential equations, the Hodgkin-Huxley equations.

Most of the students are not biomathematics or quantitative biology majors. Also, they are not experienced modelers. So, we take an approach where they learn the

basics of how to analyze and model dynamical systems without lengthy derivations or proofs. In fact, the students never even solve a system of differential equations analytically, because, as they learn, most non-linear differential equations lack analytic solutions. The approach is laid out in a wonderful textbook by Garfinkel et al. *Modeling Life* (2017 — video lectures are available for free at https://modelinginbiology.github.io/videos/). This book not only makes dynamical systems, including chaotic ones, easily understandable for Life Sciences students, but also the examples in the book are centered around physiological, ecological, and epidemiological systems rather than the usual assortment of physical systems (e.g., mass-spring, pendulum, and planetary) that form the basis of most books on the topic of dynamical systems.

The UCLA course takes place during a single ten-week quarter. In class, the course lays out the underpinnings of the dynamical systems approach. Students are given ample opportunities to work with this approach in weekly simulation laboratories that are overseen by teaching assistants who are highly skilled in the analysis and interpretation of dynamical systems. Rather than forcing students into learning a lot of coding skills while also learning about dynamical systems, the class employs an inexpensive and simple to use but powerful modeling program called Berkeley Madonna (2021) that is available for both Windows and MacOS devices. Students are assessed using a variety of methods, including 1) in-class exams to assess their understanding of dynamical systems thinking, 2) simulation laboratory exercises to assess their development as implementers, 3) interpretation of models, 4) and finally a modeling project where students develop models in areas of their own interest that put together all of what they learn.

When the course is complete, the students are able to analyze the behavior of systems of ordinary nonlinear differential equations for equilibria and, more generally, attractors, as well as qualitative changes in the dynamic behavior of these systems with changes in parameter values (bifurcations). They are able to develop differential equation models of biological systems, including the identification of relevant state variables and their feedforward and feedback interactions. In addition, they know how to use computer simulations to calculate and visualize the behavior of particular solutions to differential equation models of biological systems. Most importantly, they become proficient enough in these skills to develop and simulate *de novo* differential equations models from basic system descriptions.

Dynamical systems are the language of nature. Understanding how to describe natural systems in terms of dynamical systems and then analyze the behavior that emerges from them empowers students to go from describing what may be happening based on linear thinking about first principles, to what is actually happening and what can (and, just as importantly, cannot) emerge from the behavior of these systems. Solving 21st-century problems in biology requires contemporary quantitative thinking. Only then can we truly appreciate the importance and beauty of the geometry of nature.

MATHEW ABRAMS: TRAININGSPACE: RESOURCES FOR COMPUTATIONAL NEUROSCIENCE & NEUROEDUCATION WITHOUT BORDERS

The International Neuroinformatics Coordinating Facility (INCF) now includes TrainingSpace (https://training.incf.org/), an online hub to make neuroscience educational materials more accessible to the global neuroscience community. TrainingSpace was developed in collaboration with INCF, HBP, SfN, FENS, IBRO, IEEE, BD2K, and the iNeuro Initiative. As a hub, TrainingSpace provides users with access to:

- Multimedia educational content from courses, conference lectures, and laboratory exercises from some of the world's leading neuroscience institutes and societies.
- Four study tracks (Neuroinformatics, Computational Neuroscience, Neuroscience, and Brain Medicine) to facilitate self-guided study.
- Tutorials/demonstrations of resources (tools, software, and services) available for neuroscience research.
- Neurostars.org, a Q&A forum.
- KnowledgeSpace, a data discoverability portal/encyclopedia for neuroscience that provides users with access to over 1,600,000 files of publicly available data and models as well as links to literature references and scientific abstracts.

In addition to the subject themes of the four study tracks (neuroinformatics, computational neuroscience, brain medicine, and neuroscience), TrainingSpace also includes lectures, courses, and tutorials in computer science, data science, ethics, career development, and open science. All content objects in TrainingSpace include a general description, learning objectives/topics covered, difficulty level, and links to required software/tools and prerequisites courses/lectures. Many lectures also include downloadable lecture notes and slides and links to Jupyter notebooks, code repositories, and sample datasets. TrainingSpace also provides access to tutorials on open science resources that instructors could incorporate into their courses (instructors are free to include all multimedia content found in TrainingSpace into their courses).

To facilitate self-guided learning in TrainingSpace, INCF is pursuing its integration with Neurostars.org, a question and answer forum for neuroscience researchers, infrastructure providers and software developers. Neurostars provides access to experts from around the world for students and teachers. Sample datasets and models are available in KnowledgeSpace (https://knowledge-space.org/), which was developed jointly by INCF, the Human Brain Project, and the Neuroscience Information Framework (NIF). KnowledgeSpace is a repository of global neuroscience web resources, including experimental, clinical, and translational neuroscience databases, knowledge bases, atlases, and genetic/genomic

resources, all of which have been integrated into TrainingSpace. KnowledgeSpace also serves as an encyclopedia for neuroscience that combines general descriptions found in Wikipedia with more detailed content from InterLex, a dynamic lexicon of neuroscience concepts supported by NIF. KnowledgeSpace then integrates the content from those two sources with the latest neuroscience citations found in PubMed and data found in some of the world's leading neuroscience repositories.

CONCLUSION

Speakers in this workshop offered a variety of viewpoints. One that was widely endorsed was providing hands-on instruction, including the use of resources that allow learning with actual data sets, as described by Dr. Abrams. There were differences in opinion about how steeped in mathematical training students need to be; both Dr. Fairhall and Dr. Kass suggested the need for a fairly rigorous background, while Dr. Babiec described a course in which differential equations are not necessary. As for whether Python or MATLAB is better for implementing algorithms, both approaches are valuable depending on your students' background and which pedagogical objectives you want to achieve.

The future will no doubt provide even more complex computational models that strive to integrate levels from the molecular to the behavioral. The organizers believe that this workshop will help faculty to teach computational aspects of neuroscience at both the undergraduate and graduate levels.

Videos of the workshop can be viewed at the Society for Neuroscience's Neuronline website: https://neuronline.sfn.org/career-paths/teaching-computation-in-neuroscience. Viewing is unlimited for SfN members, and currently includes up to five free articles for others.

REFERENCES

Aljadeff Y, Lansdell B, Fairhall A, Kleinfeld D (2016) spike train analysis, deconstructed. Neuron 91(2):221–259.

American Association for the Advancement of Science (2011) Vision and change in undergraduate biology education: a call to action. Washington, DC: AAAS. Available at https://visionandchange.org/finalreport/

Anderson JR (2007) how can the mind occur in the physical universe? Oxford, UK: Oxford Press. Available at https://doi.org/10.1093/acprof:oso/9780195324259.001.0001

Bargmann C et al. (2014) BRAIN 2025: Brain Research through Advancing Innovative Neurotechnologies (BRAIN) working group report to the advisory committee to the director, NIH. Bethesda, MD: National Institutes of Health. Available at https://braininitiative.nih.gov/strategic-planning/brain-2025-report.

Berkeley Madonna, Inc. (2021) Berkeley Madonna. Berkely, CA: University of California, Berkeley. Available at https://berkeley-madonna.myshopify.com/.

Cannon WB (1929) Organization for physiological homeostasis. Physiol Rev 9:399–431.

Du Lac, C et al. (2019) The BRAIN Initiative 2.0: From cells to circuits, towards cures. Report of the NIH Director BRAIN Initiative Working Group 2.0. Bethesda, MD: National Institutes of Health. Available at https://braininitiative.nih.gov/strategic-planning/acd-working-groups/brain-initiative-20-cells-circuits-toward-cures.

Einevoll GT et al. (2019) The Scientific Case for Brain Simulations. Neuron. 102(4):735-744. Doi:10.1016/J.Neuron.2019.03.027.

Fairhall A (2014) The receptive field is dead. long live the receptive field? Curr Opin Neurobiol: 25:Ix-Xii. Doi: 10.1016/J.Conb.2014.02.001.

Garfinkel A, Shevtsov J, Guo Y (2017) Modeling life: the mathematics of biological systems. New York, NY: Springer.

Grisham W (2014) Book Review: MATLAB For Neuroscientists: an introduction to scientific computing in MATLAB (Second Edition) J Undergrad Neurosci Educ 13(1) R3-R4.

Grisham W, Lom B, Lanyon L, Ramos RL (2016) Proposed training to meet challenges of large-scale data in neuroscience. Front Neuroinform 10:28. Doi: 10.3389/Fninf.2016.00028.

Juavinett A (2020) The Self-Organized Movement to Create an Inclusive Computational Neuroscience School. Simons Foundation Blog, September 17, Available at https://www.simonsfoundation.org/2020/09/17/the-self-organized-movement-to-create-an-inclusive-computational-neuroscience-school/.

Kass RE, Eden U, Brown E (2014) Analysis of neural data. New York, NY: Springer.

Kass RE et al. (2016) Ten simple rules for effective statistical practice. PLOS Comput Biol. Available at doi.org/10.1371/journal.pcbi.1004961.

Kass RE et al (2018) Computational neuroscience: mathematical and statistical perspectives. Annu Rev Stat Appl 5:183–214

Miller G (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review 101(2) 343-352.

Nylen, EL, Wallisch, P (2017) Neural data science: a primer with MATLAB and Python, Boston, MA: Academic Press.

Pugh GE (1977) The biological origin of human values. New York, NY: Basic Books.

Wallach P, Lusignan ME, Benayoun MD, Baker TI, Dickey AS, Hatsopoulos NG (2014) MATLAB for neuroscientists: an introduction to scientific computing in MATLAB. Boston, MA: Academic Press.

Received November 8, 2020; accepted January 20, 2021.

Address correspondence to: Dr. William Grisham, Dept. of Psychology, UCLA, PO Box 95-1563, Los Angeles, CA 90095-1563. Email: wgrisham@g.ucla.edu