BayesCog: A freely available course in Bayesian statistics and Hierarchical Bayesian Modeling for Psychological Science

Lei Zhang Aamir Sohail

2025-01-01

Summary

We present a semester-long educational course (see 14-week schedule) and Jupyter Book that provides introductory training in specifying, implementing, and interpreting computational models that characterize human social behavior and neuroscience. Through readings, discussions, and labs (e.g., Jupyter Notebook tutorials using Python), students will receive hands-on training in using mathematical models to test specific theories and hypotheses and explain unobservable aspects of complex social cognitive processes and behaviors. These aspects broadly include learning from and for others, learning about others, and social influences on decision-making and mental states.

Background in introductory psychological science and basic research methods/statistics is highly recommended prior to participating in this course. Programming experience is helpful, but not required or expected. Although the course content is tailored for undergraduate students, it was developed to be easily adapted for trainees at higher career stages.

Statement of Need

Developing formalized computational models is among the most promising approaches for advancing theories in psychological science and neuroscience (**Guest2021?**), including characterizing theories regarding the neural basis of social behavior (**Stanley2013?**; **Cheong2017?**; **Jolly2017?**). This practice requires researchers to explicitly formalize scientific theories as quantifiable predictions that can be tested and falsified with empirical data. Computational models in research offer several advantages: they remove any ambiguity of verbal descriptions of theories, constrain the dimensionality of theories,

shift focus away from null hypothesis significance testing in low-dimensional experimental designs (e.g., 2 x 2 factorial designs), and provide an open and reproducible approach (via programming code) for testing formal models of a theory (**Jolly2017?**; **Guest2021?**).

Despite the promise and increased use of computational models to characterize human social behavior and neuroscience (**Zhang2019?**), few psychology programs actually offer quantitative training required for introductory level computational modeling and literacy (**Guest2021?**; **Jolly2017?**). Although summer schools (e.g., Neuromatch Academy (**VanViegen2021?**), Methods in Neuroscience at Dartmouth (MIND), Neurohackademy, Cognitive Modeling Academy), self-guided online tutorials and papers (**Wilson2019?**; **Zhang2019?**; **Lockwood2020?**), and community-oriented workshops (e.g., Brainhack Global (**Craddock2016?**; **Gau2021?**)) have filled this pedagogical gap in recent years, these programs may not reach audiences who are not familiar with computational modeling in research (e.g., undergraduates).

Therefore, we present course materials and accompanying Jupyter Book (**ExecutableBooksCommunity2020?**) to make training in computational modeling more accessible to both teachers and students in university settings and online.

Learning Objectives

Throughout this course, students are expected to meet the following goals:

- Understand the following: what are computational models, why use computational models, when to use computational models, and how models are fit to data
- Survey selected topics on social behavior through readings, discussions, and labs (e.g., Jupyter Notebook tutorials using Python)
- Learn how to write code for data science using the Python programming language and Jupyter Notebooks
- Fit mathematical models to data and evaluate their performance
- Interpret the results of social behavioral experiments
- Gain a richer understanding of how computational modeling advances psychological science and neuroscience

The course begins with readings and discussions on the utility of computational modeling in psychological science and neuroscience (see Statement of Need) and training in scientific Python (**Module 01**). Students will learn how to set up an Anaconda Environment and work in Google Colaboratory to run Python code in Executable Notebooks. They will then learn the basics of the Python programming language, including using `numpy` for scientific computing

(**Numpy?**), `pandas` for data management (**Pandas?**), and `matplotlib` for data visualization (**Matplotlib?**).

Then, students will learn the basics of modeling (**Module 02**). Using simulated data, they will learn how to fit linear and nonlinear models using least-squares minimization with a grid search, the `scipy.optimize.minimize` module (**Scipy?**), and the analytic solution for least-squares optimization. Using real data (**OConnell2021?**), they will learn how to estimate parameters (e.g., intercepts, slopes) from a linear model using both Ordinary Least Squares (OLS) and Maximum Likelihood Estimation (MLE). They will also learn how to assess model performance using R^2 and the Bayesian Information Criterion. They will compare their produced results to the actual results from the paper.

Next, students will learn the basics of reinforcement learning in non-social contexts (**Module 03**). They will learn how to run a "Two-Armed Bandit" experiment using PsychoPy (**Peirce2019?**) and explore behavioral data from the task. Following exercises adapted from (**Wilson2019?**) and implemented in Python, students will simulate learning using three models: (1) random responding, (2) noisy win-stay-lose-shift, and (3) Rescorla-Wagner reinforcement learning. They will use MLE to fit reinforcement learning models to simulated data and recover model parameters (e.g., learning rate, temperature) (**Palminteri2017?**).

During the second half of the course, students will apply what they have learned in the context of social experiences and interactions (**Lockwood2020?**; **Zhang2019?**) with an increased focus on reading empirical research papers with computational models, presenting findings and interpreting results in journal club settings, and discussing with other students. Topics include learning about threats from others, learning how actions affect the outcomes of others, theory of mind, updating beliefs about others' states and traits, and social influences on decision-making. They will complete an adapted version of a social reinforcement learning task (**Lockwood2016?**) in PsychoPy (**Peirce2019?**) and develop models in Python to evaluate and compare the likelihood of each model given real data (**Module 04**). Finally, they will develop their own project and write a research proposal (formatted using guidelines for the National Science Foundation Graduate Research Fellowship Program Award application).

All course materials were designed to be modular. For example, if students are knowledgeable in Python basics and Jupyter Notebook (**Module 01**), then teachers could begin with model fitting (**Module 02**). If students are knowledgeable in model fitting with maximum likelihood estimation (**Module 02**), teachers could start with reinforcement learning (**Module 03**). Any tutorial can be adopted, transformed, or built upon for other educational purposes (e.g., courses, single class sessions, workshops) under a Creative Commons Attribution-ShareAlike 4.0 International License. Information on how teachers can use these materials can be found in the README documentation.

During the first iteration of the course in the spring 2021, students were 54.5% seniors (4th year), 36.4% juniors (3rd year), and 9.1% sophomores (2nd year). Broadly, students reported that they enrolled in the course to learn how to use Python and computational modeling to accelerate psychological research, to learn how to work with data and research methodology beyond a traditional research methods & statistics course, to learn skills relevant to a career in data science, and to gain a different perspective on topics from other classes. Most students reported spending 3-5 hours per week on readings and assignments (one student reported spending up to 10 hours and one student reported spending less than three hours). On a scale from 1 (strongly disagree) to 5 (strongly agree), students reported that the course was academically challenging and rigorous (M=4.75, SD=0.43) and that the course materials (M=4.00, SD=1.22), course-specific technologies (M=4.00, SD=1.22), and assignments (M=4.25, SD=0.83) helped them meet the stated learning objectives of the course.

Tutorials & Exercises

Module 01 - Scientific Python

	Run	View
Tutorial: Jupyter Notebooks	Open in Colab	Open in Book
Tutorial: Python Basics	Open in Colab	Open in Book
Tutorial: Working with Data	Open in Colab	Open in Book
Exercise	Open in Colab	Open in Book

Module 02 - Introduction to Modeling

	Run	View
Tutorial: Linear Modeling	Open in Colab	Open in Book
Tutorial: Non-linear Modeling	Open in Colab	Open in Book
Exercise	Open in Colab	Open in Book

Module 03 - Reinforcement Learning

	Run	View
Tutorial: Two-Armed Bandit	Open in Colab	Open in Book
Tutorial: Models of Learning	Open in Colab	Open in Book
Exercise	Open in Colab	Open in Book

Module 04 - Social Learning

	Run	View
Tutorial: Prosocial Learning	Open in Colab	Open in Book
Exercise	Open in Colab	Open in Book

Students have the option to interact with each tutorial notebook using one of two ways. The first option entails installing Anaconda 3 on their local computers

with a custom environment made for this course and then opening each notebook using Jupyter Notebook. Teachers can walk students through the installation steps using the instructions provided on the Getting Started page. However, this can be rather difficult to implement with students' different machines, operating systems, background, etc. As a fail-safe, students also have the option to use Google Colaboratory, which runs the code on Google's cloud servers for **free**. There is a button at the top of each tutorial page that opens each Jupyter Book page in Google Colaboratory.

Once students are able to run notebooks (locally or on the cloud), they can interact with all of the code in the tutorials. Teachers should guide students through all of the "tutorials", which, for example, will allow them to demonstrate to students how tweaking certain variables in real-time would change the outputs. The "exercises" at the end of each module were created for the students to apply what they have learned. They can work on these by themselves or in groups. Links to example solutions for all exercises are provided on each page. In addition, teachers should familiarize themselves with the recommended readings associated with each module and use them to introduce students to each topic, generate discussions and ideas among students, and provide relevant background for each tutorial. For example, the Wilson & Collins (2019) paper can be assigned for reading and discussion prior to beginning any notebook tutorials for **Module 03**.

For additional background on the content covered in each tutorial (technical or theoretical), teachers and students are encouraged to review the references and/or resources listed at the beginning of each tutorial or the non-exhaustive list of resources on the Resources page. Content-related questions from students, teachers, and contributors related to the course can also be submitted directly on the GitHub repository. However, we request that individuals make a good faith effort searching for their answer in the relevant readings or resources listed at the beginning of each tutorial before submitting questions.

Future Directions

This course will introduce trainees to using computational models for characterizing human social behavior and neuroscience. However, the covered topics are not exhaustive or comprehensive. We provide a growing list of literature that extends beyond the purview of the course. Furthermore, available tutorials only cover a selected range of topics. We welcome contributions from the community to extend the breadth and depth of tutorials. These extensions may include improving or expanding existing content (e.g., stylistic improvements, alternative optimization algorithms), adding new tutorials on topics covered throughout the course, or adding new tutorials on related topics not covered throughout the course.

We appreciate any feedback on errors, omitted credit, or new ideas for this

evolving project. These can be communicated on GitHub by either reporting an issue or requesting an enhancement. All contributions will be credited and should follow the community guidelines outlined on the Contribution page.

Acknowledgments

S. A. R. is grateful to Dr. Deborah Sterns, Joscelin Rocha-Hidalgo, and the Georgetown Center for New Designs in Learning & Scholarship (CNDLS) for allowing him to use their teaching resources throughout the development of this syllabus. He is also grateful to Kelly C. Martin and Paige Amormino for reviewing initial drafts of this manuscript, Project Jupyter for making it possible to create and share these materials in a Jupyter Book (**ExecutableBooksCommunity2020?**), and all the students who participated during the first semester of this course (Spring 2021) for their insightful class contributions and feedback. The format of this course was largely inspired by the MIND Computational Summer School, Neuromatch Academy (**VanViegen2021?**), Dr. Maximilian Risenhuber's Computational Neuroscience course, Dr. Robert C. Wilson's Modeling the Mind course, Dr. Luke J. Chang's DartBrains Jupyter Book (**Chang2020?**), the research conducted in Dr. Abigail A. Marsh's Laboratory on Social and Affective Neuroscience, and the countless conversations with colleagues about the selected topics.

Author Contributions

S. A. R. designed and taught the course, generated the tutorials, assembled the Jupyter Book, reviewed the tutorials and contents of the Jupyter Book for bugs, typos, errors, stylistic enhancements, and clarity, revised contents of the Jupyter Book, and wrote the manuscript. S. A. R. learned to implement the relevant course methods through conducting research and participating in various summer courses (e.g., 2018 MIND Computational Summer School, 2018 Neuroscience School of Advanced Studies in Social Decision-Making, 2019 Computational Psychiatry Summer Course). He also received relevant pedagogical training as a Teaching Assistant (TA) for Neuromatch Academy, as a TA for Dr. Maximilian Risenhuber's Computational Neuroscience course, and as a participant in the Apprenticeship in Teaching Program at Georgetown University. The course materials were created to make the hands-on elements from these experiences as well as smaller, related workshops led by S. A. R. more accessible in university settings and online. L. G. participated in the course as a student in spring 2021, reviewed the tutorials and contents of the Jupyter Book for bugs, typos, errors, stylistic enhancements, and clarity, revised contents of the Jupyter Book, and wrote the manuscript.

References