Perspectives on Computational Modeling

MACS 30100

Meeting Days & Times:

- Blend of asynchronous and synchronous (via Zoom)
- Synchronous Sessions: Tuesday & Thursday, 8:00-8:50 am CDT
- Class Zoom Link: *TBA*

	Dr. Philip Waggoner	TA 1	TA 2	TA 3
Email	pdwaggoner@uchicago.edu	TA 1	TA 2	TA 3
Office Hours (via Zoom)	Th 9-11 am (or by email appt.)	TA 1	TA 2	TA 3
GitHub	pdwaggoner	TA 1	TA 2	TA 3

Introductory Remarks

Welcome to *Perspectives on Computational Modeling*! I am excited to walk through this material with you this quarter. Before jumping into the specifics of the course, I want to first present my approach to (re)designing and teaching this course as a core course in the MACSS *Perspectives* sequence.

My goal for you: **to develop an understanding of computational modeling that results in a well-stocked toolbox of computational methods and programming skills for solving social problems in a manner that mimics the real-world of problem solving**.

There are a lot of "buzz" words in this goal, so let me pull it apart a bit. First, the notion of a computational toolbox is captured in the scope of methods we are covering, which is vast. This is largely a survey course, not a statistical theory or computer science course, though we will draw heavily on key mathematical, statistical, and programming concepts routinely as it serves our main goal. As such, I will focus more on providing a full view of the range of computational methods available and how to efficiently implement them, rather than diving deep into a narrower subset of methods.

Re: "real-world problem solving", to minimize the all-too-common gap between information consumption, application, and knowledge creation, I have designed each element of this course, from modes of assessment to the techniques we will be covering, to mimic a real-world approach to computational problem-solving. That is, we will use computers, computational models, and group-based tasks, all of which will range from complex to simple, dependent on the scope of the problem at hand. By crowdsourcing problems and actively participating in learning, it is my hope that you complete the course and the program with a set of *learned experiences*, not just a head full of knowledge.

Further, there are many, many perspectives on all of these topics, from methods, to programming approaches, to applications of both. And to further complicate matters, I like different presentations of these topics for different reasons. As such, readings and lectures can be thought of as my distillation of the many sources I use, cite, and trust. At times it may feel awkward (e.g., blending a CS approach to a method with a statistical approach to the same method, or citing multiple readings from very different perspectives, but on

the same topic). My goal will be to marshal all of the modes of content dissemination in this class (lectures, readings, in-class work, problem sets) to unify these different approaches, giving a solid computational social science approach to addressing problems and questions in society.

A quick word on how to approach assigned readings and lectures. I assign a fair amount of readings for two reasons: first, this is a graduate school course, and the best way to learn a field or topic, is to engage in the scholarly dialogue via reading. And second, I am a firm believer that, despite how simple or complex a model or technique may be, engaging with that model or technique multiple ways and in multiple contexts will only serve to deepen an understanding of it, which will lead to more reliable, rigorous application of it. Thus, in addition to the readings assigned for each day's class, students will also need to view my recorded lecture (summation) on the material. This will not only contribute to the previous goal of hearing the information multiple times, but also serves as a guide as to the concepts I feel are most important in light of our main goal for the course. My expectation is that by hearing and interacting with these concepts in multiple arenas, both passively (readings and lectures) and actively (in-class work and problem sets), all students will be on the same footing moving forward.

To that end, implied in this expectation is that some (or possibly many) of the methods and techniques we cover will be review for some, *or* entirely new for others. My goal in presenting the selected methods in the order and context I have chosen, is to showcase these methods as core members of a modern, computational toolbox. That is, we can get much more complicated on every method covered in the course. But in order to get complicated, understanding the foundational methods first as well as how they fit together in a single computational social science framework is essential. I encourage those of you who want to go deeper and farther than we are able to go in this course to reach out to me directly at any point and let me know this. I would be happy to recommend additional and more complex readings, complex extensions of the methods we cover, point you to other courses on campus that go deeper, or even work with you directly on applying more complex versions of a given technique(s).

Many will think this is a really challenging course, whereas others may feel it's relatively simple. As this is an interdisciplinary program, field, and thus course, balancing the range of backgrounds and skills is a tricky endeavor. Thus, regardless of your background, and driving toward our main goal for the course, I assume no baseline of understanding of computational modeling for anyone in the course. Rather, I only assume that at this point in the program you have the requisite math, statistics, and general object-oriented programming skills necessary to learn and apply new methods and models.

A quick note for those who feel the course is simple: *push yourself!* If it's easy, great; make it hard. Complicate the algorithm after the simpler version. Complicate the approach to visualizing the results. Complicate the approach to presenting and discussing results, e.g., calculate and discuss hypothetical shifts across features. Complicate it with *other* methods that get at the issue a different way or are based in different assumptions, and then justify your approach. I will never penalize anyone for going beyond what I ask; just make sure that if you do, you first do what I ask, and then justify your decision for going beyond.

In sum, I expect a lot from my students. But I never ask anyone to do something I haven't done myself, and further, I make myself and the TAs widely available to help you through the process as much as possible.

Course Objectives

By the end of the course, students should be able to:

- Define a range of statistical learning modeling strategies and demonstrate their properties
- Distinguish between modeling strategies and assess their individual strengths and weaknesses
- Implement models using open source computing, such as R and/or Python
- Evaluate model performance
- Interpret model results for inference and prediction
- Visualize information and data using appropriate graphical techniques

Prerequisites: Students should have taken and successfully passed: MACS 30000: Perspectives on Computational Analysis; MACS 33001: Mathematics and Statistics for Computational Social Science, or similar training in math and statistics; MACS 30121: Computer Science with Social Science Applications 1, MACS 30500: Computing for the Social Sciences, or a similar programming course in R or Python.

Course Structure

The class will be "flipped", comprising both asynchronous work and synchronous meetings as a class via Zoom. The term flipped simply means the traditional "lecture" component is given outside of normal class meetings and is pre-recorded. Class time, then, is reserved for reiterating the high points, crowdsourcing questions and problems, and working together to apply the techniques/methods of the day.

Importantly: for those with connection issues, **please reach out to me early (and as often as makes sense) to keep me in the loop**. I will work with students experiencing these issues on a case-by-case basis. All sessions will be recorded and posted after class time for students to go back and review as/if they like.

First, the asynchronous components typically include:

1. *Readings & (watching) Lectures*: Each Tuesday and Thursday prior to class time (8:00 am CDT): complete readings for the day, watch the pre-recorded lecture, and completed the reading/lecture quiz for the day
2. *Posting Questions*: Each Tuesday and Thursday prior to class time (8:00 am CDT): post a question (or two) you had while reading or watching the lecture. This can be anything that was unclear at any level for the day's material. I will review these prior to class time, and select a few to discuss and work through as a class.

Second, the synchronous components typically include meeting at the regularly scheduled course days and times via Zoom. During these periods when we meet as a class, the session will be broken into three parts:

1. Brief refresher on the day's topic based on the recorded lectures and your reading
2. Addressing major/common questions from the readings/lectures (via posted questions prior to class)
3. Practice applying the method(s) with [*usually*] social science data in R (**note**: students may use Python or Julia, but there will be less support for these given that I conduct my classes in R, which has the added bonus of deepening your programming skills in multiple languages throughout the program (R and Python) rather than sole focus on one or the other).

The structure for the in-class sessions may vary. Some days we will work as a class the entire period. Other days, students will work in smaller groups. And still other days may be a blend. Throughout, I will ask for participation and/or call on students to share solutions or errors they encountered in an effort to crowdsource solutions and bring the class in on the "successes" and the "failures/errors." This is another way of pursuing our goal of mirroring the real world of computational problem-solving, which is rarely (if ever) done in isolation.

Each class day (Tuesday and Thursday) *before 5 pm CDT*, all code from that day's session is due. will be grading these for completion, and to keep track in general of student progress. More details on grading and these submissions below.

Though you should budget your time how you wish, a typical week in this course might look like:

1. Friday-Monday, watch the following Tuesday's lecture, do the reading, and take the quiz; while the information is fresh, post a question that came to mind while going over **Tuesday's** material
2. Show up Tuesday ready to discuss and apply the method(s) you just learned about
3. Submit your in-class code by 5 pm CDT Tuesday

4. Tuesday-Wednesday, watch the lecture, do the reading, and take the quiz; while the information is fresh, post a single question that came to mind while going over **Thursday's** material
5. Show up Thursday ready to discuss and apply the method you just learned about
6. Submit your in-class code by 5 pm CDT Thursday

A few tips for what it might be worth:

- I recommend going day-by-day, instead of working ahead, as the volume of content will pile up quickly.
- Consider taking Friday off (from my class at least; and get back to it Saturday-Monday) to get some space, clear your head, and mentally prepare for the following week. We will not only be covering a lot of material in this course, but the quarter system in general moves very quickly.
- Don't forget, you will also have several problem sets to complete over the course of the quarter as well. These will need to be worked into your weekly workflow accordingly. Though a lot, you should plan and manage your time the way that works best for you and how you work. This will likely be different for everyone. That's OK; there are many ways to do a single thing. This class will help you learn and/or strengthen time management skills in a fast-paced, intense context. Here again, we are driving at our main goal presented at the outset, which is to mirror the real world, which itself is fast-paced and extremely competitive. Therefore, an additional benefit of this course is that it serves as a relatively lower-stakes opportunity to hone your workflow and practice these skills.

Text & Materials

Required

1. Sign up for a free GitHub account at: https://github.com. Github classroom will be used to submit all problem sets. For those unfamiliar with using version control for assignment submissions, see Dr. Soltoff's excellent guide: https://cfss.uchicago.edu/faq/homework-guidelines/#homework-workflow
2. Locally download R (https://cran.r-project.org/mirrors.html) and RStudio (https://rstudio.com/products/rstudio/download/#download):
3. (ISL) James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*. New York: Springer. Free PDF version: http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf
4. (ESL) Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2012. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. Free PDF version: https://web.stanford.edu/~hastie/ElemStatLearn/
5. Kennedy, Ryan, and Philip Waggoner. 2020/forthcoming. *Introduction to R for social scientists: A Tidy programming approach*. Chapman & Hall/CRC Press. See course files on Canvas for free PDF version.
6. Waggoner, Philip. 2020/forthcoming. *Unsupervised machine learning for clustering in political and social research*. New York: Cambridge University Press. Free PDF version: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3693395

Recommended

1. VanderPlas, Jake. *Python Data Science Handbook*. O'Reilly. This is a great reference text for Python translations of many data and modeling operations we will perform in R.

Assessment

All final grades are rounded to the nearest decimal (e.g., 88.38% = 88.4%). I use the following grading scheme to determine your final grade: A (93-100), A- (90-92), B+ (88-89), B (83-87), B- (80-82), C+ (78-79), C (73-77), C- (70-72), D+ (68-69), D (63-67), D- (60-62), F (0-59). I deduct a letter grade per day any assignment is late (the "clock" starts the minute after the deadline; e.g., for assignment x due at submission time, $t = 5$, $t_x \in \{5:01, \ldots, 11:59\} = B$).

There are 5 components to final grades: (1) Daily Reading/Lecture Quizzes (20%), (2) Daily Class Participation (5%), (3) Daily in-class coding (15%), (4) Problem Sets (30%), and (5) a Final Exam (20%).

- **Daily lecture/reading quizzes** (20%): Ahead of each synchronous class session, students will complete a quiz based on the readings and lecture notes for the given class day. Except in a few cases, there will be two (2) quizzes per week. These are due prior to the respective class. In general, these will be brief quizzes with about 5-10 questions per quiz, ranging in difficulty. They will typically be embedded in the pre-recorded lectures, where completion is required to continue watching the lectures. They act as a quality and attention checks for class preparation.
- **Daily class participation** (5%): Prior to each in-person class session each week, students will post a question to course **Canvas** page based on the day's content. The question can be *anything that came to mind while reading and watching the lectures*. These must be submitted prior to the start of class at 8:00 am CDT on each synchronous class day (Tuesday and Thursday).
- **Daily class coding** (15%): After each synchronous class session, students will submit scripts/notebooks of their in-class code, which must include *at least attempts* made on each question/prompt to receive full credit. These will be graded mostly for completion, and will serve as a regular check on progress and work. This will also help to flag problematic areas that might arise, while also helping me catch and correct for my own blindspots from lectures, readings, in-class work, etc. *Note*: **a complete submission includes only a single script/notebook with no output** (i.e., just the code). In-class code is due each class day at the appropriate Canvas module (as an attachment to a "Discussion" post), prior to 5:00 pm CDT.
- **Problem sets** (30%): Throughout the course, students will complete and submit five (5) equally-weighted problem sets via **Github Classroom**. These are designed to push you to go deeper on a given topic, and to focus on building on what we have covered together in class, *not* replicate what we have don together. In most cases, this is meant to push you to practice and think creatively, which will lead to students becoming knowledge creators, not just consumers.
- **Final exam** (20%): At the end of the course, students will complete a cumulative final exam. More details to come, but there will be a study guide released the week before the exam.

Diversity & Inclusion

The University of Chicago is committed to diversity and rigorous inquiry from multiple perspectives. The MAPSS, CIR, and MACSS programs share this commitment and seek to foster productive learning environments based upon inclusion, open communication, and mutual respect for a diverse range of identities, experiences, and positions.

Any suggestions for how we might further such objectives both in and outside the classroom are appreciated and will be given serious consideration. Please share your suggestions or concerns with your instructor, your preceptor, or your program's Diversity and Inclusion representatives: Darcy Heuring (MAPSS), Matthias Staisch (CIR), and Chad Cyrenne (MACSS). You are also welcome and encouraged to contact the Faculty Director of your program.

This course is open to all students who meet the academic requirements for participation. Any student who has a documented need for accommodation should contact Student Disability Services (773-702-6000 or disabilities@uchicago.edu) and the instructor as soon as possible.

Course Outline

Note: The schedule below is tentative and may change. If changed, students will be made aware with sufficient time to adjust.

Syntax: *assignments released in italics*; **due dates in bold**

Week 1

- Jan. 12 (Tuesday) Introduction and model building
  - Reading:
    - Syllabus
- Jan. 14 (Thursday) Computational foundations: Model-based programming, Feature engineering for missing data
  - Reading:
    - Kennedy and Waggoner 2020. "Introduction to R for Social Scientists", chs. 1-4 (*skim as/if needed*)
    - Kuhn and Johnson. 2019. "Feature Engineering and Selection: A Practical Approach for Predictive Models", *skim* chs. 1 and 8, http://www.feat.engineering/

Week 2

- Jan. 19 (Tuesday) Model fitting, bias and variance
  - *PS1 Released*
  - Reading:
    - ISL ch. 2
    - ESL ch. 7.1 - 7.5
- Jan. 21 (Thursday) Resampling
  - Reading:
    - ISL ch. 5
    - ESL ch. 7.10 - 7.11

Week 3

- Jan. 26 (Tuesday) Regression problems
  - **PS1 Due to Github Classroom by 11:59 pm CDT**
  - Reading:
    - ISL ch. 3, 7.1
    - ESL ch. 3.2
- Jan. 28 (Thursday) Regression problems, cont'd
  - Reading:
    - ISL ch. 6.1, 6.2
    - ESL ch. 3.3, 3.4, 3.6

Week 4

- Feb. 2 | (Tuesday) Classification problems
  - *PS2 Released*
  - Reading:
    - ISL ch. 2.2, 4.1 - 4.3
    - ESL ch. 4.1, 4.4
- Feb. 4 (Thursday) Classification problems, cont'd
  - Reading:
    - ISL ch. 4.4 - 4.5, 3.5
    - ESL ch. 4.3, 13.3

Week 5

- Feb. 9 (Tuesday) Nonlinear estimation
  - **PS2 Due to Github Classroom by 11:59 pm CDT**
  - Reading:
    - ISL ch. 7.1 - 7.5
    - ESL ch. 5.1 - 5.5, 9.4
- Feb. 11 (Thursday) Nonlinear estimation, cont'd
  - Reading:
    - ISL ch. 7.6 - 7.7
    - ESL ch. 6.1 - 6.3, 9.1

Week 6

- Feb. 16 (Tuesday) Tree-based inference
  - *PS3 Released*
  - Reading:
    - ISL ch. 8.1 - 8.2
    - ESL ch. 9.2, 8.7, 10 (skim), 15 (skim)
    - (skim) Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. "Unbiased recursive partitioning: A conditional inference framework." *JCGS*, (2006): 651-674, https://eeecon.uibk.ac.at/~zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf
- Feb. 18 (Thursday) Support vector machines
  - Reading:
    - ISL ch. 9
    - ESL ch. 4.5, 12.1 - 12.3

Week 7

- Feb. 23 (Tuesday) Distance, clustering, partitioning
  - *PS4 Released*
  - **PS3 Due to Github Classroom by 11:59 pm CDT**
  - Reading:
    - ISL ch. 10.3

    - ESL ch. 14.3
    - Waggoner Clustering Short Book (skim)

- Feb. 25 (Thursday) Dimension reduction, manifold learning
  - Reading:
    - ISL ch. 10.2
    - Roweis, Sam T., and Lawrence K. Saul. "Nonlinear dimensionality reduction by locally linear embedding." *Science* 290, no. 5500 (2000): 2323-2326, https://science.sciencemag.org/content/sci/290/5500/2323.full.pdf?casa_token=eI9QqjO1DM4AAAAA:c2ZJfmR5vXKbVhWg6TsvVNjpHICTcmKvkhx8AYyaq19dt13coKQLcpZXnvhP3NaiRCSJjXK5eeSG6g
    - Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing data using t-SNE." *JMLR*, http://www.cs.toronto.edu/~hinton/absps/tsnefinal.pdf
    - McInnes, Leland, John Healy, and James Melville. 2018. "Umap: Uniform manifold approximation and projection for dimension reduction." https://arxiv.org/abs/1802.03426

Week 8

- Mar. 2 (Tuesday) Artificial neural networks, self-organizing maps
  - *PS5 Released*
  - **PS4 Due to Github Classroom by 11:59 pm CDT**
  - Reading:
    - ESL ch. 11.3 - 11.4, 14.4
    - (original introduction of SOM for reference only; *no need to read*) Kohonen, Teuvo. Self-organizing maps. Springer. https://link.springer.com/book/10.1007%2F978-3-642-97610-0
- Mar. 4 (Thursday) Deep learning, autoencoders
  - Reading:
    - Goodfellow et al., "Deep Learning", ch. 14 (skim), https://www.deeplearningbook.org/contents/autoencoders.html

Week 9

- Mar. 9 (Tuesday) Model-agnostic interpretation
  - **PS5 Due to Github Classroom by 11:59 pm CDT**
  - *Final exam study guide released*
  - Reading:
    - Christoph Molnar. "Interpretable Machine Learning", ch. 5, https://christophm.github.io/interpretable-ml-book/agnostic.html
    - Rudin, Cynthia. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead." *Nature Machine Intelligence* 1, no. 5 (2019): 206-215, https://arxiv.org/pdf/1811.10154.pdf
- Mar. 11 (Thursday) **Class Canceled: Study Day**
  - Reading: Final exam study guide; course notes

Week 10

- Mar. 16 (Tuesday) **Final Exam**