PS 811: Introduction to Statistical Computing

Instructor: Michael DeCrescenzo

Spring 2020

Credits: 1 (pass/fail)
Lecture: Fri. 8–9 a.m., 3218 Social Sciences
Website: `mikedecr.github.io/teaching/computing-811`

E-mail: `decrescenzo@wisc.edu`
Office Hours: Weds. 3–5 p.m.
Office: Colectivo (State St.)[1]

About This Course

Every (?) contemporary social science research project happens on a computer. We have to manage data (spreadsheets, interviews, primary sources), conduct analysis (not all of it statistical), maintain documentation about the project, organize literature reviews, create bibliographies, and write the dang paper. All of these tasks are easier with a principled understanding of *computational tools* and *reproducible workflow*.

Political Science 811 is an introduction to computational/programming techniques for political and social sciences. Much of the course is geared toward quantitative research (data science work in R), but many of the topics, tools, and skills we cover are transferable to qualitative and theoretical research as well (project workflow, reproducibility, technical document preparation, Git…). I want to emphasize that these skills are valuable for technical/research-focused careers *outside of academia* as well.

The goal of the course is to build computational skills that they don't teach you in other methods courses. A statistics course may discuss a particular model, but it may not teach you how to prepare your data or manage the model's output. A qualitative methods course may discuss a process tracing methodology, but it may not give you tools for diagramming the process with a computer graphic. Our goal is to be prepared to tackle not just our coursework but also *real projects* and all of their idiosyncratic problems.

Learning Outcomes

More specifically, the course will focus on the following skills and goals.

1. Manage a self-contained, reproducible research project.
2. Prepare, model, and visualize data using the statistical programming language **R**.
3. Create a dynamic research paper using **R Markdown**.

[1] I have chronic back pain that makes it difficult for me to hold office hours in North Hall. Look for me at either the upstairs or downstairs standing bars.

4. Track changes to plain-text source files with **Git**.

Class Setup and Bureaucratic Details

This is a 1-credit course graded on a pass/fail basis. The instructional mode is all face-to-face and follows the "Traditional/Carnegie" credit model.[2]

Our course reserves the classroom for *two hours every Friday* (8–10 a.m.). Since this is a 1-unit course, we will hold one hour of "required" instruction time (8–9 a.m.) and one hour of optional/open practice time (9–10 a.m.) for working on take-home exercises in a structured environment with nearby help. It takes practice to learn a programming skill, so I encourage you to take the second hour of practice time seriously.

The required in-class time will be a mixture lecture, code demos, and code practice. If a lesson has slides, they may contain some code examples, but they will not contain every piece of important information. Expect me to circulate scripts and other notes through Canvas.

Software

We will use a few important software tools in this course:

- **R**, a statistical programming language. If you haven't already, install it at https://cloud.r-project.org/. Unlike many statistical tools of generations past, R is actually cool. And free.
- **Rstudio**, an app that provides a development environment for powerful and efficient R programming. If you haven't already, download the free *desktop version* at https://www.rstudio.com/products/rstudio/download/. Be sure to install it *fully*.[3]
- **Git**, a version control system. Version control is like "track changes" for code; it shows you a record of changes to a file, which facilitates iterative project development and collaboration. It may already be installed on your computer. To find out, follow the installation instructions for your operating system at https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. **I do not recommend installing from source.**

We will learn R with a focus on the "Tidyverse" packages rather than built-in R ("base R"). This is an intentional pedagogical choice. The Tidyverse approach contains tools that are simultaneously powerful, flexible, and easy to use. It is a style of R programming that you are *likely to use in the future*—more likely than an all-base style, I would bet. Given that we don't have much time together, I propose we cut straight to the fun stuff.

Readings and Other Materials

Here are some free texts for the tools/skills we learn in this class, most (all?) of them open-source. The course schedule below contains links to additional resources.

[2] At least two hours of outside work for every one hour spent in class, per the university's credit policy: https://teachlearn.provost.wisc.edu/course-syllabi/

[3] This is a warning for (at least?) Mac users. When you download Rstudio, you get a folder containing an executable app. Do *not* simply run the app from within the folder that you download. Instead, copy (drag) the Rstudio app all the way into your applications folder.

R/Tidyverse:

- R for Data Science, Grolemund and Wickham (hereafter "R4DS")
- Statistical Inference via Data Science: A ModernDive into R and the tidyverse, Ismay and Kim (hereafter "Modern Dive")

R Graphics (`ggplot2`):

- ggplot2: Elegant Graphics for Data Analysis, 3rd ed., Wickham (2nd ed. on Canvas)
- Data Visualization: A practical introduction, Healy
- ggplot Flipbook, Reynolds

R Markdown:

- R Markdown: The Definitive Guide, Xie, Allaire, and Grolemund
- bookdown: Authoring Books and Technical Documents with R Markdown, Xie

Git:

- Happy Git and GitHub for the useR, Bryan, "the STAT 545 TAs," and Hester

Bonus:

- Notes from Spring 2018, DeCrescenzo (not as thorough)
- Notes from Spring 2019, Judge-Lord
- What They Forgot to Teach You About R, Bryan and Hester (hereafter "WTF-R")
- Advanced R, Wickham
- Advanced Data Analysis from an Elementary Point of View, Shalizi
- Learn R on Twitter: R-Ladies, Mara Averick, Thomas Mock, Jenny Bryan, Hadley Wickham, Gina Reynolds, Julia Silge, Kara Woo, Emily Reiderer, Alison Hill, Emi Tanaka, Andrew Heiss, and Michael Kearney.

Seeking Help

Computer programming can be frustrating because computers are stupid, and error messages are not always helpful. Fortunately, help is out there.

Take the optional "Hour 2" of class seriously. It is set aside for your benefit. It provides an opportunity to practice programming concepts (1) as they are freshly introduced in class and (2) while I am around to help with problems.

Outside of class time, I can provide help in office hours and (sparingly) by email, but try to refrain from reaching out *until you have exhausted other sources of help* such as StackOverflow or RStudio Community.[4] Why? Because finding solutions using documentation and online resources is an essential skill for learning any programming language. You will learn these tools (especially R) better by learning to help yourself! Note that the course material is designed to be self-contained enough that everything you need to complete take-home assignments should be included in course notes, texts listed above, and the resources provided as we go. Obviously, problems that are specific to your individual project needs may require solutions from beyond the course material.

[4] Tip: Google "rstats" instead of just "r"

Deliverables

In past years, teaching evaluations for this course suggested that students wished they could be held more accountable for practicing these skills on an ongoing basis. As such, we will have some assignments throughout the semester. They will be of three types.

1. Short (short!) almost-weekly assignments for routinizing basic software skills. These will be focused mainly on data tasks in R.
2. A "midterm" assignment: an R Markdown document that combines R code and writing to talk through a short data analysis process. This assignment is designed to combine data science and workflow practices.
3. Final project: an open-source, reproducible research project that integrates data analysis, writing, and version control. This does not need to be an original project that you devise *only for this class*—instead, I recommend that you "double up" this project with a paper that you already have to write for another course (such as PS 813). The difference is that I will grade the computational work that happens in the background of the project (code, project organization, etc.) rather than the scholarly work in the paper itself.[5] The project should culminate in a self-contained, reproducible project repository that features an R Markdown paper, uses code for its analysis, graphics, bibliography, etc., is version-controlled with Git, and is pushed to Github.[6]

I will provide more information about assignments as the semester progresses.

Grading and Assignment Policies

Collaboration: I encourage *good-natured* collaboration and consultation with your classmates, but you must turn in your own work. Plagiarism is surprisingly easy to spot—I have spotted it in the past—and will not be tolerated (see "Academic Integrity" below). Assignment sheets may contain additional guidance on acceptable forms of collaboration, and I am *more than happy* to answer your questions on acceptable forms of collaboration. The final project, however, should be done on your own.

Attendance: Your attendance is your own responsibility, but skipping class will harm your understanding of essential concepts. This class exists because it is challenging to learn these skills without guidance. If difficult life circumstances are regularly affecting your ability to attend class, please speak with me ASAP so we can create accommodations. For one-off absences, consult your classmates for notes.

Grading and late work: Please submit assignments (except the final project) on Canvas by 8 a.m. on their due dates, which will be Fridays. My hope is to keep these assignments short in order to minimize late work and get solutions to you ASAP. I will keep track of assignment performance using a check/plus/minus/zero system.

[5] Please consult me if you have questions or concerns about your project idea. I am flexible and happy to accommodate project ideas for students who are not enrolled in PS 813 and/or who have other project formats that could be amenable to this assignment.

[6] Or some other remote repository service like GitLab or BitBucket.

Academic Integrity

I encourage collaboration but take academic integrity and honesty very seriously. If you cheat, fabricate, plagiarize, collaborate outside acceptable limits, or help others commit these acts, you can face disciplinary action including but not limited to failing an assignment, failing the course, or facing more serious disciplinary action dispensed by higher university authorities. Substantial or repeated cases of misconduct will be referred to the Office of Student Conduct & Community Standards for additional review. For more, see https://conduct.students.wisc.edu/academic-integrity/. I am happy to answer questions about collaboration, citation practices, and more!

Other policies

Diversity and Inclusion

Diversity is a source of strength, creativity, and innovation for UW-Madison. We value the contributions of each person and respect the profound ways their identity, culture, background, experience, status, abilities, and opinion enrich the university community. We commit ourselves to the pursuit of excellence in teaching, research, outreach, and diversity as inextricably linked goals. The University of Wisconsin-Madison fulfills its public mission by creating a welcoming and inclusive community for people from every background—people who as students, faculty, and staff serve Wisconsin and the world. Visit https://diversity.wisc.edu/ for more.

Accommodations

The University of Wisconsin–Madison supports the right of all enrolled students to a full and equal educational opportunity.

Religious observance will *always* be a valid reason to request accommodations from me.

The Americans with Disabilities Act (ADA), Wisconsin State Statute (36.12), and UW-Madison policy (Faculty Document 1071) require that students with disabilities be reasonably accommodated in instruction and campus life. Reasonable accommodations for students with disabilities is a shared instructor and student responsibility. Students are expected to inform instructors of their need for instructional accommodations by the end of the third week of the semester, or as soon as possible after a disability has been incurred or recognized. Instructors will work either directly with the student or in coordination with the McBurney Center to identify and provide reasonable instructional accommodations. Disability information, including instructional accommodations as part of a student's educational record, is confidential and protected under FERPA. For more, visit http://mcburney.wisc.edu/facstaffother/faculty/syllabus.php.

Other extraordinary life circumstances can interfere with class performance. There are various resources available to help students through challenging times, including the Dean of Students Office (https://doso.students.wisc.edu/) and the Division of Student Life (https://students.wisc.edu/).

Please discuss any potential accommodations with me ASAP, including accommodations that may supersede my other class policies. It is important to me that the classroom is welcoming to all students.

Email Policies

It is virtually certain that I will email you with updates on assignments and course materials. I expect you to read and act on those updates.

I am usually quick to respond to student emails, but some warnings:

- I may not answer certain questions directly and instead refer you to the syllabus, a textbook chapter, or other materials I distribute for class.
- Last-minute emails about assignments are not guaranteed timely responses. We have at least a week to work on every assignment, and I have intentionally placed my office hours near assignment due dates, in order to give students ample opportunity to identify issues and seek help.
- I don't anticipate grading disputes (again, the course is pass/fail), but if you have questions/concerns, my policy is to discuss them in person and not over email.

Class Schedule

The schedule below contains course topics, readings, and other helpful links to check out as we go. Due dates for the "midterm" and "final" are included below, but short assignments will be assigned on an ad-hoc basis.

The schedule has *a lot* of recommended readings and resources. Don't panic. I have erred on the side of over-inclusion, and none of the readings are strictly "assigned." That said, each of them is worth skimming on one's own time. Several of them may come in handy for assignments in the near term. All of them contain lessons that can save you from headaches over the long term.

The schedule is divided into three units. The first unit will give you enough background to achieve liftoff (and complete assignments for PS 813 in R!). The second unit goes beyond the basics to teach you valuable, real-world skills that are under-emphasized (or ignored?) in a typical methods course. The final unit covers advanced skills and practices that catapult you to new frontiers of coding efficiency, statistical understanding, and market value ($$$).

Unit 1: Foundations

January 24: Introductions and software rundown.

- The Plain Person's Guide to Plain Text Social Science

January 31: Project-based workflow. Intro R Markdown.

- WTF-R: Project-Oriented Workflow and Practice safe paths
- Naming things
- Good enough practices in scientific computing

February 7: R essentials. Intro Tidyverse.

- R4DS: Workflow Basics
- WTF-R: Saving source and blank slates

- Style guides: short and long
- Base R Cheat Sheet
- Funny but weirdly sublime: aRrgh: a newcomer's (angry) guide to R and The R Inferno

February 14: Data manipulation. The pipe operator (`%>%`).

- Data Processing with dplyr & tidyr (skip `tidyr` section for now)
- R4DS: Data Transformation and Pipes
- Suzan Baert tutorials: selecting, mutating, filtering, summarizing
- Data Transformation (dplyr) Cheat Sheet
- Modern Dive: Data Wrangling

February 21: Graphics with `ggplot`.

- Graphics resources listed above
- R4DS: Data Visualization
- Data Visualization (`ggplot`) Cheat Sheet

February 28: Regression and model output.

- Modern Dive: Basic Regression and Multiple Regression
- Introduction to `broom`
- `estimatr` Cheat Sheet

Unit 2: What they don't teach you about real data and principled workflow

March 6: Horrors of real data. Reading, cleaning, and joining.

- R4DS: Data Import, Relational Data, Strings, Factors, Dates and Times
- Data Import Cheat Sheet
- Read Data with Multiple Header Rows into R
- Strings and Substitutions and Lubridate

March 13: Shaping data (wide, long, and "tidy"). Writing functions.

- Data Processing with dplyr & tidyr (the sections about `tidyr`)
- R4DS: Tidy Data and Functions
- Modern Dive: Data Importing and "Tidy" Data

March 20: Spring Break

March 27: It's about more than the math (or, Why LaTeX is good and R Markdown is gooder).

- Rmd resources listed above
- R4DS: R Markdown
- R Markdown Cheat Sheet
- R Markdown Driven Development
- For LaTeX-specific things: LaTeX Wikibook
- When your co-author insists on Word: redoc (work-in-progress)

April 3: Intro Git. Committing, ignoring, pushing to remote.

- Git for Humans
- Excuse me, do you have a moment to talk about version control?
- Happy Git and GitHub for the useR

Unit 3: The money makers

April 10 **(Midterm assignment due)**: Git skills for the real world. Branches, Gitflow, large data.

- A successful Git branching model
- Git can facilitate greater reproducibility and increased transparency in science
- Oh shit, Git!?!
- Opinionated analysis development

April 17: Functions you can pull up to: `apply`, `purrr`, anonymous, formulas.

- R4DS: Iteration and Many Models
- Purrr Cheat Sheet
- Purrr - tips and tricks
- Purrr Tutorial

April 24 and May 1: Advanced topics. Pick two.

- Simulation-based methods (bootstrapping, cross-validation, randomization inference)
- Causal vs. predictive models (causal inference, intro machine learning)
- The command line (shells, dotfiles, paths)
- Remote computing (server clusters, containers/virtual machines)
- Databases (SQL)
- Tools/skills rundown for non-academic careers (a blend of the above)
- "Advanced reproducibility" (dissertations with `Rmd`, `make`, modularity)
- Will also take suggestions

Final project due (pushed to Git remote) by Thursday, May 7, 7:45 a.m.