Computational Social Science
A Course in R

Christopher T. Kenny
christopherkenny@fas.harvard.edu

Fall 2026

Research in the social sciences is never hurt, and is frequently improved, by applying computational tools and techniques. Yet, we almost never teach necessary computational skills to social science students. This course is designed to fill that gap.

This course will teach you how to use the R programming language. You will not learn statistics in this course. Instead, you will learn how to use R to download, manipulate, and analyze data. Often, R is "taught" as a set of functions or packages to fill existing needs. This course will teach you how to think in R, so that you can solve new problems as they arise. The goal is not to give you experience with a set of tools, but to give you the skills to use any tool.

Along the way, we will cover a broader range of topics in computational social science. On day one, you will learn how to write your own functions. You will learn how to interface with online data through APIs. You will write your own reproducible research documents. You will design your own R packages.

Our assignment structure is designed to have you open R every single day. You don't have to spend a lot of time using R every day, but you should open it.

Most importantly, you will be introduced to alternative computational tools. Contrast allows us to understand the strengths and weaknesses of the tools we use. So, this class will be taught in R, but you will see examples in Python and Julia. We will not give you a comprehensive introduction to these languages, but you will see enough to understand the differences. Your assignments will be run through Quarto, but you will be exposed to LaTeX and Typst. Quarto naturally integrates with each of these, but using them directly will give you a better understanding of the trade-offs. You will learn how to use GitHub for version control and I'm not going to show you any alternatives because GitHub is the best.

This class is designed to be applicable to advanced undergraduates and graduate students in the social sciences. Graduate students will be expected to write a final project, in the form of an R package, which applies to your research. Undergraduates will also write a final project, based on designing a package, but with less emphasis on the research application. Graduate students

enrolled in a *terminal master's degree* program or professional will be held to the same expectations as any other graduate student, but will be offered more leeway in the definition of "research" for the package project.

The expectation for this class is that you've taken some course which uses R. There is no formal prerequisite, but you should be comfortable with the basics of R. If you want to take this course without any R experience, you should take a week *prior to the class* to first work through Harvard's Math Prefresher for Political Scientists, specifically Chapters 7–11.

Class Meetings

This course uses a combination of lecture classes and section classes. Lectures are larger course meetings designed to introduce new concepts and explain skills. Sections are small course meetings focused on practicing the new skills and concepts. Attendance is required for both lectures and sections.

Lecture

Lecture will meet twice weekly for 75 minutes. Lectures are designed to include a mix of lecture, discussion, and hands-on exercises. Each lecture will include an attendance quiz, which will be graded on "intelligent" completion. There will be five answer choices, but you will be graded on whether you answered one of the four plausible answers. The goal is to encourage you to attend lecture, but not to penalize you for missing a question. However, if you are not paying attention and select the non-plausible answer, you will be penalized.

Life happens. We will have 28 lectures, but you can miss 3 without penalty. You can earn up to 25 lecture points which contribute up to 10% of your final grade. If you do the math, each additional absence will reduce your final grade by four-tenths of a point. Should this be a problem, please let me know.

Section

Section meets once-a-week for 50 minutes and your attendance is required. Section aims to help you practice the skills and concepts that you learned in lecture. Each section will include a set of exercises, which you will work on in groups. While attendance will be taken, you will not be graded on your performance.

When learning a new programming language, the primary goal is to force people to practice. You don't need to get things right or get them to work. You just need to try. After you've tried, if it didn't work, you can look at the answer. But you should always try first.

Assignments

Our course grades are determined by a combination of problem sets, prelims, and a final.

Assignment	Points	Percentage
Lecture attendance	25	12.5%
Section attendance	25	12.5%
Problem Sets	70	35%
Prelim #1	20	10%
Prelim #2	20	10%
Final proposal	10	5%
Final	30	15%

Problem sets and attendance are weighted heavily. I do not like exams which control the majority of the grade and believe that they are antithetical to your education. However, I recognize the need to push you to learn the material. So, the final is worth 15% of your grade, with 5% allocated to the proposal. This is a class that you will almost surely pass if you put in some effort, but you do need to complete all pieces of the work to earn a top grade.

Problem sets

There will be `r 14 * 5` problem sets in this course. You can miss 10 of them without penalty, but your score will be computed as `min(sum(scores), 14 * 5)`. The goal is that you will open R every single work day. If you set aside 10 minutes each day, you should be able to complete the problem sets.

Each week, you will get a set of 5 problems. You can submit problems early, but you cannot submit problems late. This takes a flexible approach, where problems are released following the first lecture each week. You can submit problems at any time, but you must submit all of them by the end of the week.

The problems are designed to be challenging, but not impossible. Most can be googled, which is absolutely fine. Only in fake situations do you have to solve problems without the internet. But beware, we want you to learn, so we force a fake situation in the first prelim, where you will be expected to solve problems without the internet, in a "blue book" exam.

Once in a while, you may feel that a problem is too difficult. That's fine. Please do your best and if you are concerned, you are free to use that as one of your 10 missed problems. If you feel they are frequently too hard, please let me know.

Please work together on the problem sets, but you must write up your own solutions. You can use any resources you like, but you must cite them.

Prelim #1 (Midterm)

The first prelim will be a "blue book" exam. This will push you to solve problems without the internet. All of the problems will be things that you should know from class or the problem sets. Prelim #1 covers material from weeks 1–5.

You are welcome to bring a single 8.5x11 sheet of paper with notes. You can write on both sides. Feel free to handwrite it yourself or print it out. You will be required to turn in your sheet with your exam.

Prelim #2 (Project)

The second prelim will be a project. You will be required to implement a project which uses the skills you've learned in class. You will be provided with a *GitHub* repository which contains a set of functions. The primary assignment is to complete this package and get it to pass a set of tests.

There are no secrets or tricks. If this package passes the check and correct tests, you will get full credit. Please make sure it does that or we will be forced to give you a lower grade.

You may not work with others on this project, though you are welcome to ask questions to the course staff. The internet may and should be used, but you must cite any code you use. The project is released the day after the first prelim and is due the by 11:59pm on day of the second prelim. All skills needed to complete the project will be covered in class.

Final

The final is a project. The goal is to have you design an R project which supports your work. If you are a graduate student, this should be a project which supports your research. If you are an undergraduate, this should be a project which supports your interests. You will be required to speak with me about your project and submit a proposal.

You can work in groups of up to 3 people. The core code can be shared, but each person should contribute. Specifically, you are expected to have one vignette per person. Each vignette must indicate who wrote it. There is no requirement that the vignettes be of equal length, but they should detail a meaningful contribution to the project. Vignettes should be mostly distinct, though they may call on shared functions or code.

The final project is due on the day of the final. On the last day of class, you will be asked to present a 1 or 2 slide summary of your project. You will be graded on the quality of your project and contribution to the project.

Other Policies

Laptop policy

Please bring your laptops to class. We will be using them to work on problems and exercises. Please don't use your laptop for anything other than class work. It will be distracting to you and to others.

Attendance

Your grade incorporates attendance. By design, there is some flexibility in the attendance policy. If you should need to miss class beyond the allowed absences, please let me know. Life happens, but we need to know how to accommodate it.

Accessibility

Please let me know if you have any accessibility needs. We will work with you to ensure that you can succeed in this course. One of the beautiful things about computational approaches is that they can be adapted to meet you needs, but we need to know what they are.

Collaboration

The exchange of ideas is essential to strong academic research. You may find it useful to share sources or discuss your thinking for any of these papers with peers, particularly if you are working on similar topics. You may even read each other's drafts and provide feedback. However, you should ensure that any written work you submit, including code, is the result of your own research and work. You must cite any books, articles, websites, vignettes, etc. that you draw on. Furthermore, while you may seek the advice or help of your peers, your submissions should be your own work. Please speak to a course instructor if you have any questions about this policy.

Use of "AI", specifically Language Models (LMs):

You are welcome to use language models to help you with your work. However, you should indicate when you have done so.

The first prelim will be a "blue book" exam, where you will not be able to use the internet. This is worth more than a full letter grade, so you should be prepared to solve problems without the internet. That said, you are welcome to use language models to help you prepare for the exam. And, you are welcome to use language models to assist you in your other work.

The world of programming is changing rapidly. We need to know that you're learning, but LM tools can help you learn, when used appropriately.

Course Materials

IDEs

Examples will be provided in RStudio. As such, I recommend that you use RStudio for this course. RStudio has also released a new IDE called Positron, which is related to VSCode.

Advanced students taking this course may prefer to use a different IDE, which is fine. We will provide support for students using RStudio and Positron, but not for other IDEs. If you are having trouble with your code, please see if it works in RStudio or Positron before asking for help.

Texts

There are no required, paid texts for this course. We will make use of three free resources:

- R for Data Science ("r4ds") by Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund
- R Packages ("r-pkgs") by Hadley Wickham and Jenny Bryan
- Advanced R ("adv-r") by Hadley Wickham

If you are struggling with R, I do recommend some paid resources:

- Data Analysis for Social Science: A Friendly and Practical Introduction by Elena Llaudet and Kosuke Imai
- Quantitative Social Science: An Introduction in tidyverse by Kosuke Imai and Nora Webb Williams

On a week-by-week basis, there will be additional readings and resources provided.

Course Schedule

Each week has a list of chapters or articles to read before the first lecture. These books contain code snippets which you are encouraged to run. None of the chapters assigned are long and the marginal add of actively engaging with code is quite valuable. Many chapters will also contain a small number of exercises, which I recommend completing just after reading the chapter.

The course is broken into 3 sections. The first focuses on R above all else. This will provide the skills and practice to be able to write your first package, which you will do by the end of the class. The second section focuses on introductions to other lanuages which can be called from R. R's design allows for this which can easily improve performance. The final section focuses on typsetting technical documents in reproducible ways, which will allow you to integrate R into end products that can be shared with people who do not use R.

Section 1: Building your expeRtise

The first section of this course is dedicated to building out your R skills. Often, we learn how certain packages work, but not the standard workflows behind them. The section starts with functions and lists in R, which are vital for most advanced applications. We then build outward to talk about other types of objects in R, sharing your code, and making your code run better and faster.

In a few of these weeks, we will made direct comparisons to Python, but all readings, lectures, and sections will be in R.

Week 1: R functions and lists

Readings

1. r-pkgs, Ch. 2: System setup
2. r4ds, Ch. 25: Functions
3. adv-r, Ch. 2: Vectors
4. adv-r, Ch. 6: Functions

Case studies

- `purrr`: a package which extends R's function and vector capabilities

Readings

Week 2: R data structures

Readings

1. r-pkgs, Ch. 1: The Whole Game

2. adv-r, Ch. 2: Names and values
3. r4ds, Ch. 20: Spreadsheets

Case studies

- `tibble`: offers a modern replacement for dataframes with improved default behaviors, like better printing
- `sf`: extends the `tibble` class to work with geospatial data

Week 3: R packages and documentation

Readings

1. r-pkgs, Ch. 3: Package structure and state
2. r-pkgs, Ch. 9: DESCRIPTION
3. r-pkgs, Ch. 16: Function documentation
4. r-pkgs, Ch. 17: Vignettes

Case studies

- `apportion`: a simple package which computes apportionments from population numbers with few dependencies and functions
- `gt`: a table-focused packages with a huge number of dependencies and functions

Week 4: Types and testing

Readings

1. adv-r, Ch. 12: Base Types
2. adv-r, Ch. 4: Subsetting
3. adv-r, Ch. 13: Testing basics
4. adv-r, Ch. 14: Designing your test suite

Case studies

- `testthat`: a popular testing library which uses itself to test its internal functions
- `ThemePark`: a pop culture library of `ggplot2` themes, which includes basic examplse of visual testing

Week 5: Custom types

Readings

1. adv-r, Ch. 13: S3
2. adv-r, Ch. 15: S4
3. `vctrs`: S3 vectors

Case studies

- `palette`: A custom class for printing and plotting color palettes.

Week 6: Data from the internet

Readings

1. `httr2`: Get started
2. r4ds, Ch. 24: Web Scraping
3. `httr2`: Wrapping APIs

Case studies

- `congress`: interfaces with the Congress.gov API to collect data
- `bskyr`: interfaces with Bluesky Social and allows for collecting data and creating new posts and messages

Week 7: Profiling and debugging

Readings

1. adv-r, Ch. 22: Debugging
2. adv-r, Ch. 23: Measuring Performance
3. adv-r, Ch. 24: Improving Performance

Case studies

- `profvis`: a package which provides an interactive visualizer for profiling R code

Week 8: Parallelization

Readings

1. `future`: A Future for R: A Comprehensive Overview

Case studies

- `wru`: uses parallelization to speed up data processing

Week 9: Version control, code review, and continuous integration

Readings

1. r-pkgs, Ch. 20: Software development practices
2. Happy Git and GitHub for the useR, Ch 1: Why Git? Why GitHub?
3. Happy Git and GitHub for the useR, Ch 9: Personal access token for HTTPS
4. Blog Post: How we review code at Pew Research Center

Case studies

- `usethis`: a package which automates many of the tasks associated with package development
- `gh`: a package which interfaces with the GitHub API to automate tasks like creating issues and pull requests

Section 2: Wrapping other languages

The second section of this course introduces you to using R as a wrapper around other languages. R is a friendly language to learn, but farms out high performance operations to other languages. The section provides a simple introduction to C++ with Rcpp, Rust with extendr, and Julia with JuliaCall.

Week 10: Rcpp

Readings

1. Extending R with C++: A Brief Introduction to Rcpp
2. Rcpp for everyone, Ch. 1-3
3. adv-r, Ch. 25

Case studies

- `redistmetrics`: uses Rcpp to iterate over large matrices

Week 11: Julia

Readings

1. JuliaCall for Seamless Integration of R and Julia:

Case studies

- `bigsimr`: uses JuliaCall to simulate large vectors of data

Week 12: Rust

Readings

1. `extendr` User Guide: A Complete Example
2. Using Rust code in R packages

Case studies

- `ymd`: A Rust-based package for dates

Section 3: Reproducible Typsetting

The final section of this course introduces you to reproducible typsetting tools. R is a friendly language to learn, but farms out high performance operations to other languages. We first introduce Quarto, the successor to RMarkdown which allows you to run code and produce PDFs, websites, and more. Then, we look at the behind-the-scenes tools used to produce PDFs, LaTeX and Typst. We will focus on using LaTeX and Typst *with* Quarto.

Week 12: Quarto

Readings

1. r4ds, Ch. 28: Quarto
2. r4ds, Ch. 29: Quarto formats
3. Quarto: Creating a Website

Case studies

- `andrewheiss/ath-quarto`: Andrew Heiss's Quarto website

Week 13: LaTeX

Readings

1. Blog Post: Creating Quarto Journal Article Templates

Case studies

- `apsr`: An APSR submission template built in LaTeX and Quarto

Week 14: Typst

Readings

1. Quarto: Typst Basics
2. Blog Post: Making Pretty PDFs with Typst (and Quarto)

Case studies

- `tufte`: A Tufte-style document built in Typst and Quarto