Sociol 703: Modern Plain Text Computing

Kieran Healy

Duke University

kieran.healy@duke.edu

About this course

This course is an introduction to modern plain-text methods of data analysis, data management, and coding. It is required for first-year graduate students in the Sociology department. It introduces a suite or "stack" of computing tools and techniques that are widely used in the department, the discipline, across the social sciences, and beyond. We will learn to use these tools and also learn why they exist and why they are important for producing work that is reliable, reproducible, and open to inspection.

Motivation

Researchers depend on computer software to get their work done. But often, they do not know enough about how their computers work. This makes their lives more difficult. I don't mean it's a shame not everyone is a software engineer ready and able to write applications from the ground up. Rather, as researchers working with data of various kinds, we just don't make the best use of our computers. Nor are we encouraged to reflect on why they work the way they do. Instead we end up fending for ourselves and picking things up informally. Or, instead of getting on with the task at hand, course instructors are forced to spend time bringing people up to speed about where that document went, or what a file is, or why the stupid thing stopped working just now. In the worst case, we never get a feel for this stuff at all and end up marinating in an admixture of magical thinking and sour resentment towards the machines we sit in front of for hours each day.

All of that is bad. This course is meant to help. While the coding and data analysis tools we have are powerful, they are also kind of a pain in the neck. For the most part they are made to allow us to know what we did. They can be opened up to have their history and inner workings examined if needed. This runs against the grain of the devices we use most often—our phones—which do not work in that way at all. As a rule, apps on your phone hide their implementation details from you and do not want you to worry too much about where things are stored or how things are accomplished or what happens if you need to do the same thing again later. They do that for very good reasons. But it does mean that even if you use a powerful computer constantly, as we almost all do now, it does not give you much of a grip on how more technical computing works. To the contrary, it makes it look strange and annoying and deliberately confusing.

The fragmented, messy, and multifacted tasks associated with scholarly research make heavy demands on software. Most of them have to do with the need for *control* over what you are doing, and especially the importance of having a *record* of what you did that you can revisit and *reproduce* if needed. They also need to allow us to track down and diagnose errors. Because our research work is fragmented and messy, this can often be a tricky process to think clearly about and work through in a systematic way.

To help address these challenges, modern computing platforms provide us with a suite of powerful, modular, specialized tools and methods. The bad news is that they are not magic; they cannot do our thinking for us. The good news is that they are by now very old and very stable. Most of them are developed in the open. Many are supported by helpful communities. Almost all are available for free. Nearly without exception, they tend to work through the medium of explicit instructions written out in plain text. In other words they work by having you write some code, in the broadest sense. People who do research involving structured data of any kind should become familiar with these tools. Lack of familiarly with the basics encourages bad habits and unhealthy attitudes among the informed and uninformed alike, ranging from misplaced impatience to creeping despair.

Reading

Required reading from books and articles will be provided on the course website Canvas or (in most cases) will be freely available online. Useful texts to acquire in hardcopy or to bookmark include:

- Hadley Wickham, Garrett Grolemund, and Mine Çetinkaya-Rundel, *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*, Second. (Sebastopol, CA: O'Reilly Media, 2023), https://r4ds.hadley.nz.
- Jeroen Janssens, *Data Science at the Command Line*, Second. (O'Reilly Media, 2021), https://jeroenjanssens.com/dsatcl/.
- Chester Ismay and Albert Y. Kim, *Statistical Inference via Data Science* (CRC Press, 2019), https://moderndive.com.
- Scott Chacon and Ben Straub, *Pro Git*, Second. (Apress, 2014), https://git-scm.com/book/en/v2.
- Kieran Healy, *Data Visualization: A Practical Introduction* (Princeton: Princeton University Press, 2019), http://socviz.co/.
- Jeffrey E F Friedl, *Mastering Regular Expressions*, 3rd ed. (Sebastopol, CA: O'Reilly Media, 2006).
- Will Landau, "The {Targets} R Package User Manual," 2022, https://books.ropensci.org/targets/.
- Hadley Wickham and Jennifer Bryan, *R Packages: Organize, Test, Document, and Share Your Code*, Second. (O'Reilly Media, 2023), https://r-pkgs.org.
- Garrett Christensen, Jeremy Freese, and Edward Miguel, *Transparent and Reproducible Social Science Research* (Berkeley: University of California Press, 2019).

Software

We will do all most of our work class using Unix-style command line tools, R, and RStudio. R is a freely-available programming language that is designed for statistical computing and widely used across the natural and social sciences, as well as in the world of "data science" generally. RStudio is an integrated development environment, or IDE, for R, a kind of control center from which you can manage the engine-room of R itself. It is also freely available.

Course website

The course website is at https://mptc.io. Each week, slides, readings, and other class material will be posted there along with additional examples and the problem set due the following week.

Weekly Schedule

Week	Topic
Week 1	Overview: Doing data analysis properly
Week 2	R, RStudio, and Quarto
Week 3	Your Computer: The file system; the terminal; the Unix way of thinking
Week 4	The Shell: Finding, listing, and inspecting things
Week 5	Editing Text: Text editors; slicing and dicing; regular expressions
Week 6	Version Control: Git and GitHub; knowing what you did
Week 7	Understanding the Network: Servers, websites, and APIs
Week 8	How R thinks
Week 9	Getting your data in and out of R
Week 10	Tabulation, grouping, summaries
Week 11	Graphs and the grammar of graphics
Week 12	Programming: writing, documenting, and testing your code
Week 13	Reproducibility and build systems: make, targets, renv

Course policies

- Attendance is required, and important. I am a reasonable person; if you need to be absent please *let me know in advance* insofar as that is possible.
- Do the assigned readings in advance of class.
- Submit problem sets, or other assignments, on time.

Required work and grading

Weekly **Class Participation** (30% of final grade) and **Problem Sets** (50% of final grade) will let you reflect on the reading and practice your skills. Problem sets are due by end of day the *Monday* after they are assigned. A **Final Project** (20% of final grade) will allow you to put what you've learned to use.

Duke community standard

Like all classes at the university, this course is conducted under the Duke Community Standard. Duke University is a community dedicated to scholarship, leadership, and service and to the principles of honesty, fairness, respect, and accountability. Citizens of this community commit to reflect upon and uphold these principles in all academic and nonacademic endeavors, and to protect and promote a culture of integrity. To uphold the Duke Community Standard you will not lie, cheat, or steal in academic endeavors; you will conduct yourself honorably in all your endeavors; and you will act if the Standard is compromised.