Teaching Computational Social Science for All

Jae Yeon Kim*, Yee Man Margaret Ng[†]

R&R at *PS: Political Science & Politics*

Abstract

Computational methods have become an integral part of political science research. However, helping students to acquire these new skills is challenging because programming proficiency is necessary, and most political science students have little coding experience. This article presents pedagogical strategies to make transitioning from Excel, SPSS, or Stata to R or Python for data analytics less painful and more exciting. First, it discusses two approaches for making computational methods accessible: showing the big picture and walking through the workflow. Next, a step-by-step guide for a typical course is provided through three examples: learning programming fundamentals, wrangling messy data, and communicating data analysis.

*Corresponding author. Assistant Research Scholar, SNF Agora Institute, Johns Hopkins University.
jkim638@jhu.edu

[†]Assistant Professor, Department of Journalism, University of Illinois Urbana-Champaign.
ymn@illinois.edu

A precise definition of computational social science remains elusive. However, a widely used definition is that it is an interdisciplinary field in which computational tools and techniques are applied to advance social science research (Salganik 2019, xviii). The key idea is that computational tools and techniques empower social scientists, including political scientists, to conduct innovative analyses of unconventional data sources on an unprecedented scale (Alvarez 2016; Grimmer, Roberts, and Stewart 2021). Finding patterns in large-scale unstructured text (Grimmer and Stewart 2013; Gentzkow, Kelly, and Taddy 2019), image (Won, Steinert-Threlkeld, and Joo 2017; Torres and Cantú 2021), or audio (Knox and Lucas 2021) data was once immensely costly and time consuming. Computational tools and techniques have made the collection (e.g., web scraping) and analysis (e.g., machine learning) of these data easier, faster, and reasonably accurate. By complementing existing quantitative and qualitative methodologies, computational methods present many new opportunities to address key questions in political science (Clark and Golder 2015; Monroe et al. 2015; Brady 2019), from polarization and misinformation in the United States (Tucker et al. 2018) to censorship and collective action in China (King, Pan, and Roberts 2013).

Opportunities also come with challenges. For political science students, learning computational tools and techniques is challenging because proficiency in programming, to which they have little prior exposure, is necessary. Most political science students choose the field because they have a deep interest in politics but not necessarily data analysis. In addition, these students first approach data analysis via integrated statistical software packages, such as Microsoft Excel, Stata, or SPSS. These tools have both a point-and-click user interface and an easy-to-learn command syntax. For students, learning a programming language, such as R or Python, could be a completely foreign skillset.

This article introduces pedagogical strategies to make this transition less painful and more exciting. Collectively, we have taught computational social science at both graduate and undergraduate levels in semester-long courses and short workshops for several years. Based on this extensive experience, we first present two principles: showing the big picture

and walking through the workflow. We then apply these principles to the design of a course that aims to make computational methods accessible.

This discussion of the practical aspects of teaching computational social science comes with two caveats. First, a computational social science project should start with a theoretically driven research question rather than the latest computational technique. Collecting the largest possible volume of social media data or applying the most powerful machine learning technique does not directly add value to political science research. These tools become relevant when they provide a new understanding of key theoretical questions in the relevant literature. In addition, linking computational methods to key political science topics has pedagogical value. It enables students to discover how they could apply these methods to explore their research interests (Williams et al. [2021](#)).

Second, emphasizing the value of data-intensive social science research does not undermine the importance of careful human investigation. Big data is not the same as good data. A researcher still needs to assess whether the data can answer their research question regarding measurement and internal and external validity (Meng [2018](#)). Because of this, computational methods are better applied when researchers are well-trained in research design and methods. A strong foundation in descriptive analysis, hypothesis testing, statistical modeling, and causal inference remains critical to answering empirical research questions.

However, having a strong motivation or a deeper understanding of a substantive problem does not automatically turn a research idea into a research project. A successful student-designed research project involving computational methods requires an adequate understanding of programming and its implementation. Nevertheless, programming is challenging for political science students because most have little prior experience. This challenge is especially acute among students from disadvantaged backgrounds because they have had relatively poor access to high-quality quantitative and computational education (for a review, see Xie, Fang, and Shauman [2015](#)). This article tackles this problem by presenting principles and specific steps for lowering the barriers to learning these exciting new tools and techniques.

1 Principles of Teaching Computational Methods

An instructor can help students learn computational methods faster and easier by showing them the big picture and workflow before getting into specific technical details. The big picture is the broad patterns regarding the beginning (input) and end (output) of specific tasks. The workflow is the logical steps that connect these two dots. For example, suppose an instructor aims to teach digital data collection, i.e., data collection and parsing from websites, Portable Document Format (PDF) files, and social media posts. The big picture here is the transformation of semi-structured data, which do not look like a spreadsheet, into structured data, which look like a spreadsheet. Without contextualizing these data collection techniques in a broad framework, the instructor could jump into introducing the latest packages and libraries for digital data collection. This approach might work if most students have experience with these techniques and their primary goal is updating their skills. However, if students have little experience with digital data collection, they might not see the forest for the trees. For them, learning so many techniques simultaneously might be confusing and overwhelming.

Because of these obstacles, it is worthwhile to take a step back and remind them of the big picture and workflow. After that, the instructor can dive into each step. Let's take web scraping as an example. What is the goal of this activity? The term web scraping provides an indication, but not a clear understanding, of the input and output data. If students scrape a website, they will obtain semi-structured data, which is a mix of HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. Even though they look messy and lack a spreadsheet-like structure, they have other kinds of structures, such as key-value pairs. Keys are labels, and values are associated information. Then, the instructor will move on to explaining the workflow. Students can leverage these structures to locate information (Step 1), extract them (Step 2), and bind these elements together (Step 3). Because students usually want to summarize, model, and visualize this information, saving these elements as a data frame akin to a spreadsheet can be effective (Step 4).

Not only does this approach help students feel less intimidated by the subject, but it serves the long-term educational goal of making students effective self-learners. Self-learning skills are valuable because in a fast-evolving field such as computational social science, cutting-edge tools are continually changing. If the hidden context—the big picture and workflow—is clarified, students find it easy to learn new techniques. Furthermore, learning this big picture and workflow helps students build a mental framework for digital data collection. For example, if students learn how to scrape websites by understanding the underlying big picture and workflow, they can quickly learn how to parse PDF files and scrape social media posts because these cases are other variants of semi-structured data.

Once students have learned the workflow, they should be taught how to combine these individual components (Perkins 2010). Typing a command is easy. Coming up with a logical sequence of computational actions is difficult. Helping students to develop an awareness of how one step is connected to another is critical because the end goal is to use these components all together. In addition, there exist considerable differences between learning each component and the methods for combining them. For example, a student might want to scrape a newspaper website to collect articles published during a specific period. However, there is usually a limitation on how many articles can be displayed on a single page. This is a useful feature that saves time for loading a webpage but makes scraping it a challenge. Manually copying, pasting, and tweaking the code on these pages is highly inefficient because in some cases, the number of pages is in the hundreds or thousands. A better alternative is learning how to repeat the iterative process used for loop and functional programming. This approach helps them write a code with only a few lines, and it works regardless of the number of pages.

This example illustrates how helping students learn an efficient workflow kills two birds with one stone. On the one hand, it helps students translate a component of their research design (collecting data from websites) into computational terms (Brooker 2019). On the other hand, it provides students opportunities to learn a computational concept—automating

iterations—in a practical context.

2 Applications

This section focuses on applying these principles to the design of a semester-long computational social science course. The following three examples are used: learning programming fundamentals, wrangling messy data, and communicating data analysis.

At the fundamental level, these principles could apply to undergraduate lectures as well as graduate seminars. However, depending on the target audience and their backgrounds and interests, the scope and focus should change. For example, if students are highly motivated and have experience in quantitative analyses, they could quickly absorb a large amount of new information. Otherwise, the instructor should slow down the pace and provide more concrete and intuitive examples. In addition, we highly recommend instructors make all the slides and code examples used in the class available to students before class so they can follow along with live in-class coding examples more easily.

2.1 Case 1: Programming Fundamentals

Introducing programming fundamentals early is the key to lowering the initial barrier and helps students to be effective self-learners. Political scientists do not need to be statisticians. However, an adequate understanding of statistics is essential for choosing the correct application and interpretation of these methods. The same applies to computational methods. Students can get immediate programming help by searching the internet. To this end, there are many useful websites such as Stack Overflow.¹

R and Python are both object-oriented and functional programming languages.² Objects

1. <https://stackoverflow.com/>

2. There are also differences between the two languages. If a student's goal is to learn to program and use it for general purposes, learning Python would be helpful because it is widely used across academia and industry. In contrast, R is developed for doing data science (Chambers 2020) and is popular among social scientists. Thus, if a student's goal is to use the power of programming for social science research, R might be a better choice. Whether Python or R is more useful depends on a student's desired career path, learning

define the actors, and functions define their behaviors. Students create, manipulate, and combine objects and functions when performing data analysis in R or Python. Programming is not magic. It is the art of logically combining objects and functions. Moving from Excel, SPSS, or Stata to R and Python means transitioning from pointing and clicking to creating and manipulating objects and functions. Consider importing a comma-separated values (CSV) file in R or Python. First, a student needs to think about what function they should use and whether the function is available without importing an outside package. Second, the student needs to provide the input (the file path for the CSV file) and save the output to an object they have named. Even this simple exercise involves understanding and using objects and functions.

Therefore, it is wise to devote the first couple of weeks to helping students to fully grasp these underlying computational concepts before teaching them serious applications. As Buchler (2009, 527) argued, a “move on, and try to figure it out from the context” learning method does not work for subjects that require skill progression. Students can copy, paste, and run example codes for machine learning without comprehending the process. The fact that their codes run without error and produce the desired output does not necessarily mean that they fully understand the process. This emphasis on fundamental computational concepts applies even to students who have extensive experience conducting quantitative analyses using SPSS or Stata. There are hurdles in moving from SPSS or Stata to R or Python. When a user creates a summary table of descriptive statistics or regression models in SPSS or Stata, knowing the object-oriented programming language is not required. In contrast, in R and Python, everything a user creates is an object. Every function takes and returns particular types of objects. Therefore, one needs to pay careful attention to how input and output data types vary. Helping students to cultivate this new habit of mind enables them to avoid unnecessary fear and anxiety.

When teaching these concepts, it is crucial to increase students’ understanding by con-

goal, and primary usage.

necting what they know to what they do not know. For example, students may know various data types, but not in a computational sense. For example, they may know spreadsheets but not necessarily data frames. Spreadsheets and data frames both have a fixed number of rows and columns. However, using font color or highlighting as data could be helpful in a spreadsheet but not in data frames. This additional information would be lost when the spreadsheet is imported as a data frame into R or Python (Broman and Woo [2018](#)).

2.2 Case 2: Wrangling Messy Data

After addressing programming fundamentals, an instructor should motivate students by demonstrating how programming can increase research productivity. Unless students are enthusiastic about programming, learning it is not the goal. Thus, they should be shown why learning programming is worth the investment of their time and effort.

Data wrangling illustrates the benefits of learning programming because almost every project requires it. In quantitative methods courses, the example data are often precooked. When students start working on real-world data, they soon discover that data are messy. Eighty percent of data analysis involves data wrangling (Dasu and Johnson [2003](#)). Although students can perform data cleaning in Excel, they need to do it manually in a point-by-point fashion. This is very labor intensive and time consuming when managing large files or cleaning multiple files. More troubling is the irreproducibility of this point-and-click approach. Instead, using programming makes data wrangling reproducible and more efficient.

The tidy data principles proposed by Wickham ([2014](#)) are an extension of Codd ([1990](#))'s relational algebra. They illustrate the big picture of data wrangling in statistical and computational analysis. First, the input data need to be a data frame (i.e., a fixed number of rows and columns) to facilitate data summarization, modeling, and visualization. Second, columns are variables, and rows are observations. Third, each observational unit comprises a data frame Wickham ([2014](#), 4). The recognition of this endgame enables students to see the specific problems with their messy data and consider using R or Python to fix them. This is

easy because the “tidyr” (Wickham 2021) and “dplyr” (Wickham et al. 2021) packages in R and “pandas” library in Python (McKinney et al. 2011) provide comprehensive tools for implementing this foundational framework.

To increase students’ awareness of the problems, wrangling can be categorized on the basis of the issues with the data. For example, one category of problems is the shape of the dataset. Many spreadsheets are stored in a wide format to facilitate inspection of the data by human eyes. If these are administrative data collected yearly from 2000 to 2010, the dataset might have columns titled with these years. Unfortunately, summarization, modeling, and visualization require a long format. To reshape the data, instructors should guide students to create the year variable representing all of these years.

While learning the typology of data wrangling problems and solutions, students should not lose sight of the broad context. Because of the many data wrangling techniques, it is easy for students to become lost in the details and fail to see the big picture. The tidy data principles are valuable because data have an entire production cycle. Raw data have little value and are not self-explanatory. Humans need to summarize, model, and visualize raw data to add value for other humans.

2.3 Case 3: Communicating Data Analysis

Teaching data wrangling and visualization poses different challenges. Data wrangling has a clear goal, as summarized in the tidy data principles. Data visualization does not. There are some abstract principles, such as the creation of accurate, easily interpretable, and aesthetically pleasing plots (Tufte 1985). However, the implementation of these principles is contextual. The decision to employ a line or bar plot or logarithmic rescaling of the y-axis depends on the information to be communicated and the target audience.

When teaching data visualization, the focus should be delivering key messages effectively. Teaching how to create point, bar, or line plots with popular packages and libraries in R or Python is insufficient. Students need to know the types of visualization that are appropriate

for delivering various forms of data. For example, representing raw data points with a scatter plot is valuable because descriptive statistics summaries (e.g., the mean) could overlook the unique distribution of the dataset (Anscombe 1973; Healy and Moody 2014).

After understanding the purpose of each visualization technique, students can learn to build the visualization of their choice from scratch. They will then learn the relationship between data and graphics. This will enhance their understanding of the variables that are used for the aesthetics (e.g., x-axis and y-axis) of a plot and their ability to add and subtract information as needed to facilitate the audience’s interpretation. This data visualization approach, which has been referred to as the grammar of graphics (Wilkinson 2005), has been extensively used in academia and industry because of the wide acceptance of the ggplot2 library in R (Wickham 2006, 2010).

2.4 Optional Subjects

Teaching digital data collection, computational text analysis, and machine learning is optional. These are powerful tools for collecting and analyzing complex data (e.g., text, images, and audio) at scale. However, teaching programming fundamentals and these advanced topics in the same course can be overwhelming. If this is to be done, the students should be made aware that these topics are optional. For example, suppose a student is interested in how public opinion shifted during social movements (e.g., Black Lives Matter) using social media data. The student can focus on a specific part of the research project (e.g., scraping social media posts) and apply the skills they learned from the course.

There are specific tips to help students learn these advanced subjects more easily. First, provide glossaries of computer science terms. Many canonical works in natural language processing, machine learning, and big data were developed in computer science. The terms often used in these works are confusing to political science students. For instance, the word features in machine learning is, in fact, variables in the social sciences. Weights and biases in deep learning are learnable parameters, and they should not be confused with their homonym

counterparts in statistics.

Second, provide mathematical or statistical intuitions. For instance, typical survey data have a larger number of observations (participants) and a smaller number of variables (questions). In contrast, typical text data have a larger number of variables (words) and a smaller number of observations (documents). Analyzing this kind of high-dimensional data poses two challenges. Researchers need to find the best ways to represent the complex data numerically because computers understand numbers, not text. Second, they should know how to reduce the number of these variables (also called features and dimensions) to ensure the analysis is driven by signals rather than noise. Making this hidden context explicit helps students better appreciate the purposes, assumptions, and mechanics of these techniques.

2.5 Assessment

To provide quick, frequent feedback, instructors should give students many small assignments instead of one big one. Designing small exercises can be problematic. If the exercises are not challenging, the students will not learn new information. If they are too complicated, students can lose confidence. To hit the sweet spot, some principles need to be observed. First, the exercises should be based on the content covered in the lecture. Second, instructors should encourage students to apply their skills beyond the contexts addressed in class. Third, the exercises should have an accompanying step-by-step guide and a template (e.g., R Markdown or Jupyter notebook). In essence, students need to be provided with a framework that they can use, opportunities that they can explore, and constraints within which they can engage in exploration without excessive risk (Wickham [2015](#)).

If instructors choose to require final projects instead of examinations, these small assignments can enable students to gradually develop necessary skills. These assignments can enhance students' motivation to extend their best effort on each assignment and, in the final project, demonstrate their ability to integrate the skills that they acquired during the semester.

When providing feedback, it is crucial to focus on how students can improve. Students should be taught to write code so that they can fail quickly, often, and systematically. Computational methods are valuable not because they prevent human error but because they make mistakes transparent. In addition, identifying mistakes in a structured way is easy in programming. Students can create tests that check whether their functions produce the expected outcomes. When grading and commenting on each assignment, it is helpful to remind students of the programming standards they need to meet, point to areas in which they need to improve, and suggest concrete steps they can take to progress their skills.

3 Concluding Remarks

The path to mastering computational methods is a journey. Learning a wide range of complex technical skills and practicing them takes substantial time and effort. When students feel left behind, it is easy for them to identify themselves with their current failures (Dweck [2006](#)). This perception discourages their efforts and widens the achievement gap between them and their peers. This article introduced pedagogical strategies that could restore these students' confidence and help them discover the joy of learning programming languages and their practical value. Understanding the big picture and workflow makes this journey easier and faster by connecting their intuitions with new information and shaping their efforts to be more structured and focused.

References

- Alvarez, R Michael, ed. 2016. *Computational Social Science: Discovery and Prediction*. Cambridge University Press.
- Anscombe, Francis J. 1973. “Graphs in Statistical Analysis.” *The American Statistician* 27 (1): 17–21.
- Brady, Henry E. 2019. “The Challenge of Big Data and Data Science.” *Annual Review of Political Science* 22:297–323.
- Broman, Karl W, and Kara H Woo. 2018. “Data Organization in Spreadsheets.” *The American Statistician* 72 (1): 2–10.
- Brooker, Phillip D. 2019. *Programming with Python for Social Scientists*. Sage.
- Buchler, Justin. 2009. “Teaching Quantitative Methodology to the Math Averse.” *PS : Political Science and Politics*, 527–530.
- Chambers, John M. 2020. “S, R, and Data Science.” *Proceedings of the ACM on Programming Languages* 4 (HOPL): 1–17.
- Clark, William Roberts, and Matt Golder. 2015. “Big Data, Causal Inference, and Formal Theory: Contradictory Trends in Political Science?: Introduction.” *PS : Political Science Amp; Politics* 48 (1): 65–70.
- Codd, Edgar F. 1990. *The Relational Model for Database Management: Version 2*. Addison-Wesley Longman Publishing.
- Dasu, Tamraparni, and Theodore Johnson. 2003. *Exploratory Data Mining and Data Cleaning*. John Wiley & Sons.
- Dweck, Carol S. 2006. *Mindset: The New Psychology of Success*. Random House.

- Gentzkow, Matthew, Bryan Kelly, and Matt Taddy. 2019. "Text As Data." *Journal of Economic Literature* 57 (3): 535–74.
- Grimmer, Justin, Margaret E Roberts, and Brandon M Stewart. 2021. "Machine Learning for Social Science: An Agnostic Approach." *Annual Review of Political Science* 24.
- Grimmer, Justin, and Brandon M Stewart. 2013. "Text As Data: the Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21 (3): 267–297.
- Healy, Kieran, and James Moody. 2014. "Data Visualization in Sociology." *Annual Review of Sociology* 40:105–128.
- King, Gary, Jennifer Pan, and Margaret E Roberts. 2013. "How Censorship in China Allows Government Criticism But Silences Collective Expression." *American Political Science Review* 107 (2): 326–343.
- Knox, Dean, and Christopher Lucas. 2021. "A Dynamic Model of Speech for the Social Sciences." *American Political Science Review* 115 (2): 649–666.
- McKinney, Wes, et al. 2011. "Pandas: A Foundational Python Library for Data Analysis and Statistics." *Python for High Performance and Scientific Computing* 14 (9): 1–9.
- Meng, Xiao-Li. 2018. "Statistical Paradises and Paradoxes in Big Data (I): Law of Large Populations, Big Data Paradox, and the 2016 US Presidential Election." *The Annals of Applied Statistics* 12 (2): 685–726.
- Monroe, Burt L., Jennifer Pan, Margaret E. Roberts, Maya Sen, and Betsy Sinclair. 2015. "No! Formal Theory, Causal Inference, and Big Data Are Not Contradictory Trends in Political Science." *PS: Political Science & Politics* 48 (1): 71–74.
- Perkins, David. 2010. *Making Learning Whole: How Seven Principles of Teaching Can Transform Education*. John Wiley & Sons.

- Salganik, Matthew J. 2019. *Bit By Bit: Social Research in the Digital Age*. Princeton University Press.
- Torres, Michelle, and Francisco Cantú. 2021. “Learning to See: Convolutional Neural Networks for the Analysis of Social Science Data.” *Political Analysis*, 1–19.
- Tucker, Joshua, Andrew Guess, Pablo Barberá, Cristian Vaccari, Alexandra Siegel, Sergey Sanovich, Denis Stukal, and Brendan Nyhan. 2018. *Social Media, Political Polarization, and Political Disinformation: A Review of the Scientific Literature*. Technical report. Hewlett Foundation, March.
- Tufte, Edward R. 1985. “The Visual Display of Quantitative Information.” *The Journal for Healthcare Quality* 7 (3): 15.
- Wickham, Hadley. 2006. “An Introduction to ggplot: An Implementation of the Grammar of Graphics in R.” *Statistics*.
- . 2010. “A Layered Grammar of Graphics.” *Journal of Computational and Graphical Statistics* 19 (1): 3–28.
- . 2014. “Tidy Data.” *Journal of Statistical Software* 59 (10): 1–23.
- . 2015. “Teaching Safe-Stats, Not Statistical Abstinence.” *Online Supplement Discussion of “Mere Renovation Is Too Little Too Late: We Need to Rethink Our Undergraduate Curriculum From the Ground Up” By G. Cobb, the American Statistician* 69.
- . 2021. *tidyr: Tidy Messy Data*. R package version 1.1.3.
- Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021. *dplyr: A Grammar of Data Manipulation*. R package version 1.0.5.
- Wilkinson, Leland. 2005. *The Grammar of Graphics*. Springer.

- Williams, Unislawa, Robert Brown, Marilyn Davis, Tinaz Pavri, and Fatemeh Shafiei. 2021. “Teaching Data Science in Political Science: Integrating Methods With Substantive Curriculum.” *PS : Political Science Politics* 54 (2): 336–339.
- Won, Donghyeon, Zachary C Steinert-Threlkeld, and Jungseock Joo. 2017. “Protest Activity Detection and Perceived Violence Estimation From Social Media Images.” In *Proceedings of the 25th Acm International Conference On Multimedia*, 786–794.
- Xie, Yu, Michael Fang, and Kimberlee Shauman. 2015. “STEM Education.” *Annual Review of Sociology* 41:331–357.