Computational Thinking for Social Scientists

Jae Yeon Kim

2021-03-04

Contents

Chapter 1

Hello World

```
print("Hello, World!")
```

```
## [1] "Hello, World!"
```

> Make simple things simple, and complex things possible. - Alan Kay

This is the website for *Computational Thinking for Social Scientists*. This book intends to help social scientists think computationally and develop proficiency with computational tools and techniques to research computational social science. Mastering these tools and techniques not only enables social scientists to collect, wrangle, analyze, and interpret data with less pain and more fun, but it also let them work on research projects that would previously seem impossible.

The book is not intended to be a comprehensive guide for computational social science or any particular programming language, computational tool, or technique. For a general introduction to computational social science, I recommend Matthew Salganik 's Bit By Bit (2017).

The book is currently divided into two main subjects (fundamentals and applications) and eight main sessions.

1.1 Part I Fundamentals

1. Why computational thinking
2. Best practices in data and code management using Git and Bash
3. How to wrangle, model, and visualize data easier and faster
4. How to use functional programming to automate repeated things
5. How to develop data products (e.g., packages and apps)

1.2 Part II Applications

6. How to collect and parse semi-structured data at scale (e.g., using APIs and web scraping)
7. How to analyze high-dimensional data (e.g., text, image) using machine learning
8. How to access, query, and manage big data using SQL and Spark

The book teaches how to do all of these, mostly in **R**, and sometimes in **bash** and **Python**.

- Why R? R is free, easy to learn (thanks to `tidyverse` and RStudio), fast (thanks to `Rcpp`), runs everywhere, **open** (16,000+ packages; counting only ones available at CRAN), and has a growing massive and inclusive community (`#rstats`).
- Why R + Python + bash?

  > "For R and Python, Python is first and foremost a programming language. And that has a lot of good features, but it tends to mean, that if you are going to do data science in Python, you have to first learn how to program in Python. Whereas I think you are going to get up and running faster with R, than with Python because there's just a bunch more stuff built in and you don't have to learn as many programming concepts. You can focus on being a great political scientist or whatever you do and learning enough R that you don't have to become an expert programmer as well to get stuff done." - Hadley Wickham

  – However, this feature of the R community also raises a challenge.

  > Compared to other programming languages, the R community tends to be more focused on results instead of processes. Knowledge of software engineering best practices is patchy: for instance, not enough R programmers use source code control or automated testing. Inconsistency is rife across contributed packages, even within base R. You are confronted with over 20 years of evolution every time you use R. R is not a particularly fast programming language, and poorly written R code can be terribly slow. R is also a profligate user of memory. - Hadley Wickham

  – RStudio, especially the tidyverse team, has made heroic efforts to amend the problems listed above. Readers will learn these recent