Course Introduction

Julia for Empirical Economics

Kosuke Shimamoto

2026-07-20

Introduction

Instructor

- Kosuke Shimamoto, a fourth-year PhD student in economics
- Research interests: IO (antitrust economics)
- I used R for four years before switching to Julia three years ago.
  - I use Julia for my own projects and GA work.
  - I can write Python, MATLAB, and SQL.
  - I have very limited experience with Stata.

Motivation

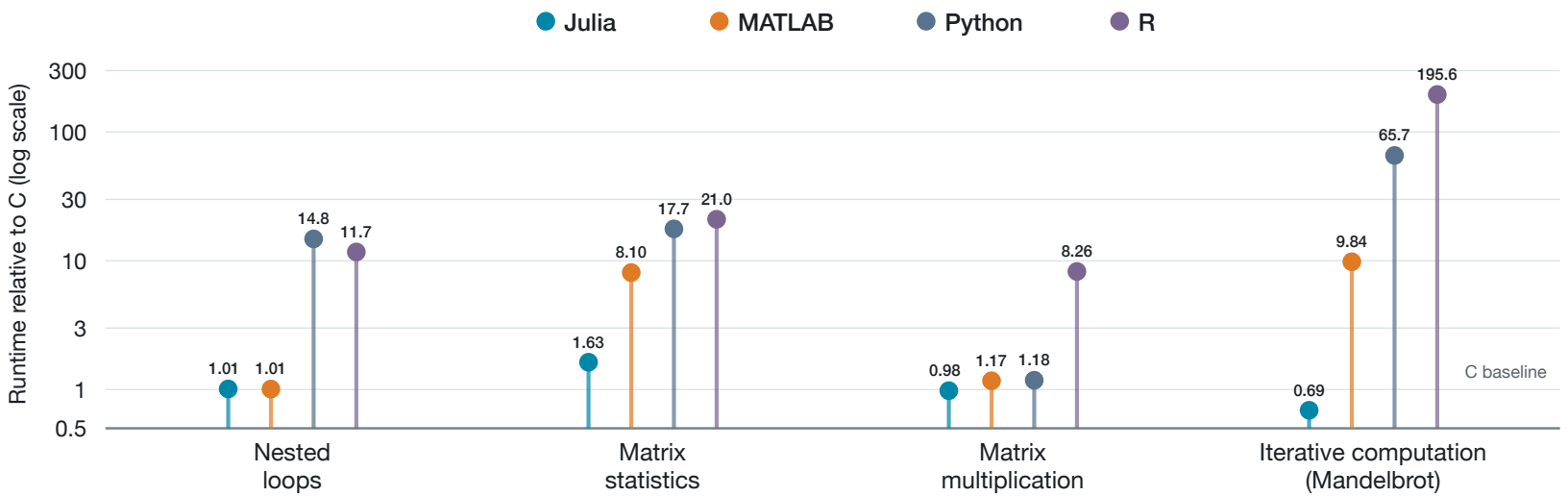
Why Julia?

- Structural estimation is increasingly common in applied economics.
- Julia combines two key features for complex estimation:
  - Fast
  - Easy to read and write
- That's all!

“Why We Created Julia”

> **“In short, because we are greedy.”**
>
> *[…]*
>
> We want a language that’s open source, with a liberal license. We want **the speed of C** with the dynamism of Ruby. We want a language that’s homoiconic, with true macros like Lisp, but **with obvious, familiar mathematical notation like Matlab**. We want something **as usable for general programming as Python**, **as easy for statistics as R**, as natural for string processing as Perl, **as powerful for linear algebra as Matlab**, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.
>
> — Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman, *Why We Created Julia* (2012)

Computational speed



Official Julia Micro-Benchmarks · identical algorithms; compile time excluded · Julia 1.0.0 · MATLAB R2018a · Python 3.6.3 · R 3.5.0 · C/gcc 7.3.1 · julialang.org/benchmarks/

- Julia's advantage is clearest in user-written code.
- A counterargument: LLMs can write code in any language, so why not use C?

Insertion sort: C

```c
#include <stddef.h>

void insertion_sort(double *a, size_t n) {
    for (size_t i = 1; i < n; ++i) {
        double key = a[i];
        size_t j = i;

        while (j > 0 && a[j - 1] > key) {
            a[j] = a[j - 1];
            --j;
        }
        a[j] = key;
    }
}
```

Insertion sort: Julia

```julia
function insertion_sort!(a)
    for i in 2:length(a)
        key = a[i]
        j = i - 1

        while j >= 1 && a[j] > key
            a[j + 1] = a[j]
            j -= 1
        end
        a[j + 1] = key
    end
    return a
end
```

- Julia code is highly readable, which is critical when reviewing LLM-generated code.

Should we use AI? Yes—or more precisely, we have to…

(Recommendations for economics PhD students on adapting their research and skill investments to AI)

> **You should be actively thinking about, and planning around, this.**
>
> When research becomes more productive, the returns to specific skills shift—so your human-capital investments should shift too.
>
> **You should be learning how to use these tools effectively.**
>
> If AI expands what researchers can produce and you do not keep pace, you may become worse off even in absolute terms.
>
> — Isaiah Andrews, *Some Thoughts on AI and Research* (2026), p. 2

- This module integrates AI tools as much as possible.
- In my opinion, Julia remains the best choice even when using AI.

This module

Learning Goals

- Write and read Julia fluently.
- Understand what Julia, related tools, and computational techniques can do.
- Use that understanding to give effective instructions to AI.
- Use AI to conduct reduced-form analysis quickly.
- Use AI to implement complex structural estimation procedures accurately and quickly on your own.

Course Principles

- This module is designed to help students independently develop and implement their own empirical economics projects entirely in Julia.
- This module is based on the idea that the most effective way to learn a programming language is to implement things yourself, even if you use extensive support from LLMs.
- The lectures aim to give students a high-level understanding of how to use Julia and related computational techniques.
    - I will move relatively quickly to introduce the main ideas across many topics.
    - I will not cover every detail of the programming language in lecture.
    - This is because you can use LLMs to study many implementation details on your own.

Course Principles

- To complement the lectures, this module assigns many hands-on exercises.
- This module strongly encourages the use of LLMs. But of course, you remain responsible for their outputs.

Syllabus

- Review the syllabus together.

Final Project

- The final project asks you to complete a full Monte Carlo simulation on your own.
- Base it on (1) your own project, (2) a paper you choose, or (3) a paper I recommend.
- I plan to create a shared Dropbox folder for everyone's PDFs and code.
  - Seeing how others work and reading their code will help us learn from one another.
  - Please make your work shareable!
  - I will show my own work in the last lecture.
- I will upload details about the required outputs and grading criteria.
- In the meantime, start thinking about what you want to do.

A few more comments…

- This module includes many of my own opinions.
  - We still do not know the best practices for using AI in research.
  - Offering my own perspective is my comparative advantage over AI.
  - Please share your ideas and techniques with me and the class!
- Your feedback matters!
  - This is a brand-new module.
  - This is my first course as an instructor.

Introductions

- Name?
- Research interests?
- Course expectations or summer highlights?