julia> Lecture One: Intro to Julia

Course: Computational Bootcamp
Name: John Higgins
Date: June 17, 2024

julia> Today's goals

1. Understand what computation is and how it relates to what we do in economics
2. Learn the basics of Julia
3. Have fun!

julia> What is computation in Economics?

- Any time you do math on a computer, you are computing!
- Some computation you might have already done before:
  - Create new variables in a dataset
  - Analyze data with built in functions
- Other kinds of computation:
  - Optimize a function (i.e. find maximum or minimum)
  - Approximate an equation without a closed form solution
  - Solve dynamic systems of equations

julia> *Why* do we need computation in Econ?

1. Enlarges the set of questions we can tackle
2. Improves the quality of our answers

> If you don't do computation, you can use theory and proofs
>> Even then, computation can guide and motivate your theory

> If you only know how to use Stata, you can work with data
>> Working with data requires some (or a lot of) computation
>> What if you want to build a structural model?
>> What if the function or package you need doesn't exist?

> In both cases, your research will be limited without strong computational skills

julia> What does this mean for you?

- Computation will be an important part of your job:
  - In your own work
  - Dealing with co-authors and colleagues
  - Reading and evaluating the work of others
- So, you should learn how to do computation well now!
- If you are a second year, your goal is to:
  1. Figure out what questions you are interested in
  2. Acquire the tools (+data!) you need to answer those questions
- This process will likely continue after the second year

julia> How do you do computation well?

Your code should be:

1. Fast
   > How long does it take to write your code?
   > When will my code finish running?
2. Readable
   > Can other people understand your code?
   > Can the future version of you understand your code?
3. Reproducible
   > Does your code give the correct answers?
   > How easy would it be to change/extend your code?

julia> How do you do computation well?

> These things are all related:

1. If your code is not readable: it is harder to find mistakes, so it is not reproducible
2. If your code is not reproducible: it is difficult to make iterative progress, so it is not fast
3. If your code takes 10 years to finish running (not fast), then there's no need to read it or reproduce it (useless)

> Luckily, there is good software available to make this easier:

> Julia and Visual Studio Code for programming
> Git and Github for version control

julia> Why Julia?

- Pros:
  - Faster runtime than basic Matlab, Python, R, Stata
  - Easier to learn and work in than C++ and Fortran
  - More readable than optimized Matlab or Python code
  - Open source unlike Matlab
- Cons:
  - C++ and Fortran can have faster run-times (sometimes)
  - You might already know other languages (switching costs)
  - Julia is relatively new and relies on public goods provision, so less packages and legacy code is available
  - Error messages can be a bit puzzling initially

julia> My advice

1. You should learn either Stata, R, or Python well
   - > For data cleaning and basic analysis
   - > Use large library of packages and legacy code
   - > Stick with what you already know best
2. You should learn Julia well
   - > For high performance computing needs
   - > This camp and Econ 880 (Computational, formerly 899) will help with this
3. Learn and use other coding languages as needed
   - > Each new language is easier to learn than the last
   - > Example: I use Python for scraping/data work, R for data, Julia for model solving/estimation, and Matlab (when I'm forced to)