julia> Lecture Two: Performance Optimization and Advanced Julia

Course: Computational Bootcamp
Name: John Higgins
Date: June 19, 2024

julia> Today’s goals

1. Understand how your computer works and interacts with programming languages
2. Learn some advanced Julia concepts, especially to improve how fast your code runs

julia> Machine language

- Computers are very good at performing calculations and executing algorithms
- Unfortunately, a computer’s hardware only understands things written in machine language
- A machine language consists of instructions written using only of binary 0’s and 1’s
- This machine language cannot easily be understood by humans

julia> Programming Languages

- Programming languages act as the translator between you and your computer
- You write a set of instructions in a programming language, such as Julia
- When you execute your code, these instructions get translated to machine language
- Your computer executes the task
- How fast your code runs will be impacted by:
    1. How you write your code
    2. How your instructions get translated to machine language
    3. What computer you are using

julia> Compiled vs. Interpreted Languages

- Programming languages generally fall into two groups:
  1. Compiled languages (C, Fortran)
     - All of the code gets translated before execution
     - Generates very efficient machine code
     - Cannot code interactively or make changes on the fly
  2. Interpreted languages (Stata, R, Python, Matlab)
     - Translates each line of code just before its execution
     - This allows for dynamic and interactive coding
     - Will not generate efficient machine code

julia> Just-in-time compilation

- Julia tries to have the best of both worlds with just-in-time compilation
- When executing a piece of code, such as a function, for the first time Julia compiles it to machine language
- When executing the next time, Julia machine language is called directly instead of re-compiling
- This allows for the benefits of interactive coding of an interpreted language
- Efficient machine code can also be generated if you follow best practices:
  - https://docs.julialang.org/en/v1/manual/performance-tips/

julia> Optimal Investment Problem

- Given capital k, we have resources

$$y = k^{\alpha} + (1-\delta)k$$

- We live forever (yay!) and discount the future at rate β. Utility from consumption is $\log(c)$
- We need to decide how much to consume (c) and how much capital to save for the next period (k')

$$V(k) = \max_{c,k'} \log(c) + \beta V(k')$$

$$c + k' = k^{\alpha} + (1-\delta)k$$

julia> Solving for V

- We know that this is a contraction mapping, so we can iterate over V until convergence
- We proceed as follows:
  1. Guess V_1. Solve the following:

     $$V_2(k) = \max_{c,k'} \log(c) + \beta V_1(k')$$

  2. Then, repeat this again:

     $$V_3(k) = \max_{c,k'} \log(c) + \beta V_2(k')$$

  3. We know that $V_n \to V$ as $n \to \infty$