

Macroeconomic Modeling with Julia

Pearl Li

Federal Reserve Bank of New York

December 18, 2017

Thank yous

- Cristina Griffa and workshop organizers
- DSGE team at the New York Fed
- John Stachurski, for much of the content on these slides
- You all for being here

Team

- **Abhi Gupta**: Senior Research Analyst at New York Fed
- **Pearl Li** : Senior Research Analyst at New York Fed

Goals and Assumptions

Assumptions

- Attendees are programmers but new to Julia
- Interested in macroeconomics

Goals

- Overview of Julia and comparisons to other languages
- Lower fixed costs to learning Julia
- Resources for further study

Schedule

Day 1

- Introduction to Julia (Pearl)
- Julia syntax (Abhi)
- Types and multiple dispatch (Pearl)

Day 2

- Julia for economists – packages and libraries (Abhi)
- State-space routines (Abhi) and DSGE.jl (Pearl)

Resources

http://github.com/FRBNY-DSGE/IADB_2017_Workshop

More resources listed there.

Software Options

Install

1. Julia
2. Packages (IJulia, QuantEcon.jl, DSGE.jl)
3. IDE if you like (such as Juno)

Or

1. JuliaPro

Or don't install

1. JuliaBox (juliabox.org) TODO

Scientific Computing

Using computation to solve scientific/engineering problems.

Tasks:

- Solve numerical problems
- Manipulate data
- Simulate
- Visualize results with tables and graphs
- Explore

And we often want to do these things **fast**.

Scientific Computing

Using computation to solve scientific/engineering problems.

Tasks:

- Solve numerical problems
- Manipulate data
- Simulate
- Visualize results with tables and graphs
- Explore

And we often want to do these things **fast**.

Speed is nontrivial

To get maximum speed, we need:

- Optimal use of hardware
- High level of control over operations/logic

First best: **assembly language**/machine code. This gives us instructions at the CPU level, specific to your chip architecture.

For example: $1 + 2$

1 + 2

```
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
movl $1, -12(%rbp)
movl $2, -8(%rbp)
movl -12(%rbp), %edx
movl -8(%rbp), %eax
addl %edx, %eax
movl %eax, -4(%rbp)
movl -4(%rbp), %eax
popq %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
```

Now imagine programming a 3-equation DSGE model like this...

You would have to optimize for specific hardware:

- Pipelining
- Cache hierarchies
- Branch prediction
- Multiprocessing
- etc, etc, etc

And change it every time there's a new processor.

Now imagine programming a 3-equation DSGE model like this...

You would have to optimize for specific hardware:

- Pipelining
- Cache hierarchies
- Branch prediction
- Multiprocessing
- etc, etc, etc

And change it every time there's a new processor.

Tradeoffs

Low level languages give us:

- Speed
- Fine-grained control
- Less machine time, more programmer time

High level languages give us:

- Abstraction (from hardware, pointers, etc)
- Automation of some tasks (garbage collection)
- Natural language representations
- Less programmer time, more machine time

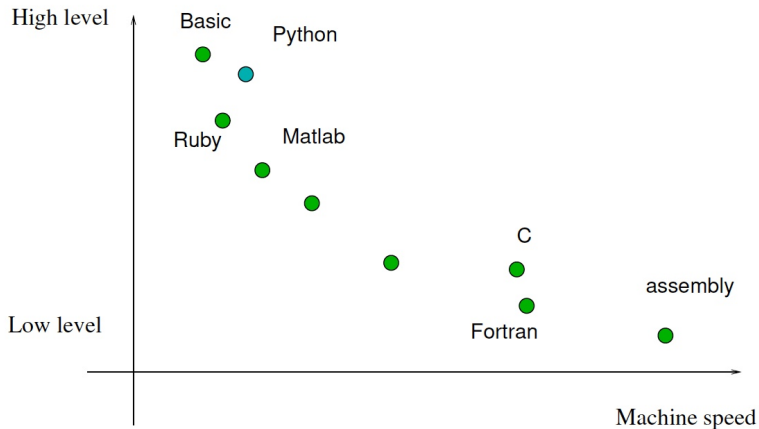
Tradeoffs

Low level languages give us:

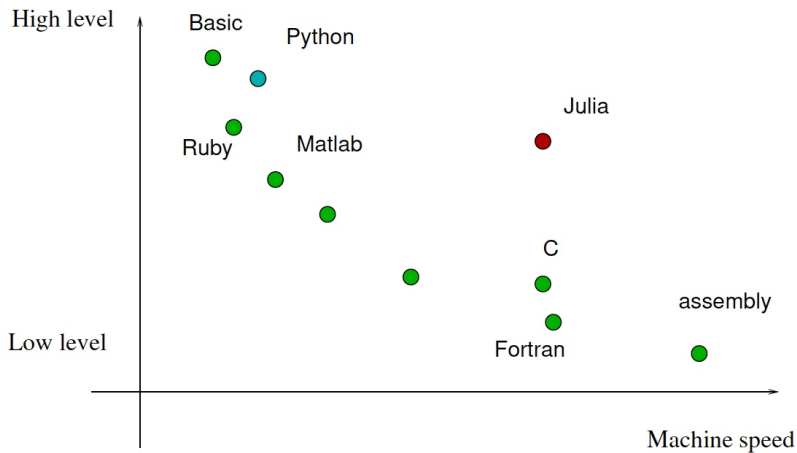
- Speed
- Fine-grained control
- Less machine time, more programmer time

High level languages give us:

- Abstraction (from hardware, pointers, etc)
- Automation of some tasks (garbage collection)
- Natural language representations
- Less programmer time, more machine time



But the curve is starting to shift...



A Horse Race

Task:

1. Compute X_1, X_2, \dots, X_n according to $X_{t+1} = \beta + \alpha X_t + W_{t+1}$, where $W_t \sim N(0, 1)$
2. Calculate and return $\frac{1}{n} \sum_{t=1}^n X_t$

Where $n = 10^7$

Julia Overview

High-level, open-source, scientific computing language

Strengths

- High productivity
- And high performance!

Potential Negatives

- Still under rapid development (will slow w/ v1.0)
- Lots of advanced features either out there or in development: rabbit holes abound

Julia Overview

High-level, open-source, scientific computing language

Strengths

- High productivity
- And high performance!

Potential Negatives

- Still under rapid development (will slow w/ v1.0)
- Lots of advanced features either out there or in development: rabbit holes abound

Why (free and) Open Source?

- Reproducibility with minimal headache:

Let's be clear: the work of science has nothing whatever to do with consensus. Consensus is the business of politics. Science, on the contrary, requires only one investigator who happens to be right, which means that he or she has results that are variable by reference to the real world. In science consensus is irrelevant. What is relevant is reproducible results.

- Michael Crichton

- Free! Don't rely on expensive licenses

Why (free and) Open Source?

- Reproducibility with minimal headache:

Let's be clear: the work of science has nothing whatever to do with consensus. Consensus is the business of politics. Science, on the contrary, requires only one investigator who happens to be right, which means that he or she has results that are variable by reference to the real world. In science consensus is irrelevant. What is relevant is reproducible results.

- Michael Crichton

- Free! Don't rely on expensive licenses

Interacting with Julia

Options:

- REPL + text editor (Atom, Sublime, Emacs, etc)
 - ▶ Minimal difference from language to language
 - ▶ Customize your text editor as much as you want
- IDEs like Juno
 - ▶ Mouse support: buttons, etc
 - ▶ Possibly most familiar for MATLAB IDE users
- Jupyter notebooks

Interacting with Julia

Options:

- REPL + text editor (Atom, Sublime, Emacs, etc)
 - ▶ Minimal difference from language to language
 - ▶ Customize your text editor as much as you want
- IDEs like Juno
 - ▶ Mouse support: buttons, etc
 - ▶ Possibly most familiar for MATLAB IDE users
- Jupyter notebooks

Interacting with Julia

Options:

- REPL + text editor (Atom, Sublime, Emacs, etc)
 - ▶ Minimal difference from language to language
 - ▶ Customize your text editor as much as you want
- IDEs like Juno
 - ▶ Mouse support: buttons, etc
 - ▶ Possibly most familiar for MATLAB IDE users
- Jupyter notebooks

Jupyter Notebooks

Browser-based front-end for Python, Julia, R, and increasingly more

- Easy to run in the cloud or on servers
- Allows for rich text and graphics - nice for presenting notes alongside computation

Examples: <http://notebooks.quantecon.org/>

Let's try one!

Jupyter Notebooks

Browser-based front-end for Python, Julia, R, and increasingly more

- Easy to run in the cloud or on servers
- Allows for rich text and graphics - nice for presenting notes alongside computation

Examples: <http://notebooks.quantecon.org/>

Let's try one!