Macroeconomic Modeling with Julia

2017 RBA Workshop

March 2017

Thanks

- Adam Cagliarini and the RBA
- Marco del Negro and the FRBNY
- Julia Computing
- Sponsors of QuantEcon

- NumFOCUS

Team

Pearl Li

- Research Analyst at the FRBNY

Erica Moszkowski

- Research Analyst at the FRBNY

John Stachurski

- Academic, interests in computation, stochastic modeling

Pablo Winant

- Bank of England, macroeconomist, lead author of dolo

Schedule

Lecture 1

- Introduction and overview (John)

Lecture 2

- The Julia language (Pearl)

Lecture 3

- Julia for economists – libraries and features (Pablo)

Lecture 4

- DSGE modeling with Julia (Erica)

Aims and Assumptions

Assumptions

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

Aims

- Background, overview and comparisons
- Lower fixed costs to getting started
- Provide resources for further study

Resources

Workshop homepage:

- https://github.com/QuantEcon/RBA_RBNZ_Workshops

Further resources listed there...

Software Options

Install

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

Or

1. JuliaPro

Overview of Scientific Computing

Tasks

- Solve numerical problems
- Produce figures and graphs
- Manipulate data
- Explore (simulate, plot, visualize, etc.)

And sometimes we need **speed**

QuantEcon

Overview of Scientific Computing

Tasks

- Solve numerical problems
- Produce figures and graphs
- Manipulate data
- Explore (simulate, plot, visualize, etc.)

And sometimes we need **speed**

The Need for Speed

Maximum speed:

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

First best = **assembly** / machine code

- Individual instructions at the CPU level

For example, let's add 1 + 2

The Need for Speed

Maximum speed:

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

First best = **assembly** / machine code

- Individual instructions at the CPU level

For example, let's add 1 + 2

The Need for Speed

Maximum speed:

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

First best = **assembly** / machine code

- Individual instructions at the CPU level

For example, let's add $1 + 2$

```
.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 a heterogeneous agent model with 5 state variables...

And then optimizing for specific hardware

- pipelining
- cache hierarchies
- branch prediction
- coprocessors
- etc.

And then Intel brings out a new processor...

Now imagine a heterogeneous agent model with 5 state variables...

And then optimizing for specific hardware

- pipelining
- cache hierarchies
- branch prediction
- coprocessors
- etc.

And then Intel brings out a new processor...

Now imagine a heterogeneous agent model with 5 state variables...

And then optimizing for specific hardware

- pipelining
- cache hierarchies
- branch prediction
- coprocessors
- etc.

And then Intel brings out a new processor...

Conclusion: There's a **trade off**

Low level languages give us

- **speed**
- **fine grained control**

High level languages give us

- **abstraction**
- **automation** of some tasks
- **natural language** representations

Conclusion: There's a **trade off**

Low level languages give us

- **speed**
- **fine grained control**

High level languages give us

- **abstraction**
- **automation** of some tasks
- **natural language** representations

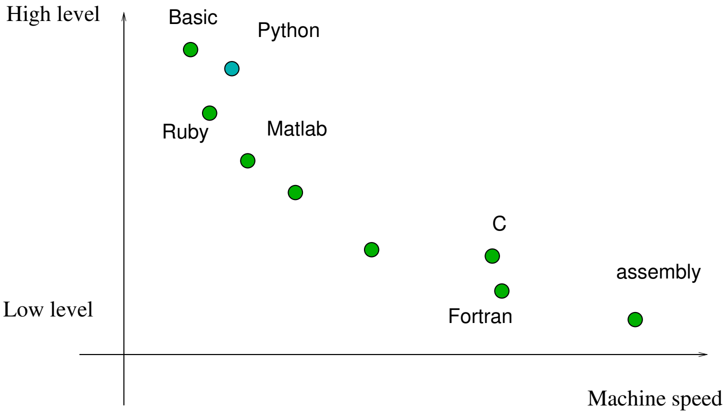
Conclusion: There's a **trade off**

Low level languages give us

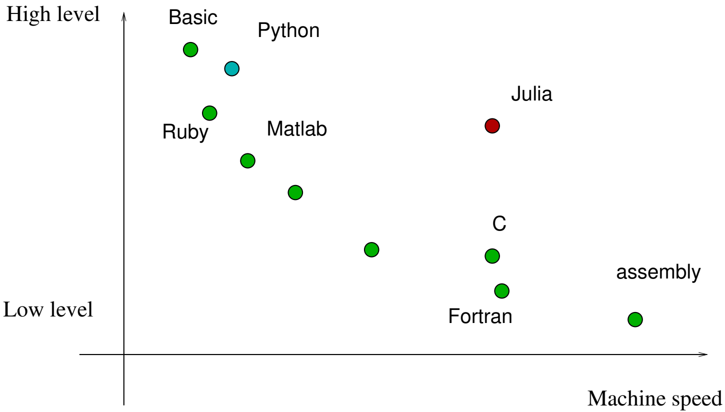
- **speed**
- **fine grained control**

High level languages give us

- **abstraction**
- **automation** of some tasks
- **natural language** representations


High level
Low level
Basic
Python
Ruby
Matlab
C
assembly
Fortran
Machine speed

That said, the curve is starting to **shift**...


High level
Low level
Machine speed
Basic
Python
Ruby
Matlab
Julia
C
Fortran
assembly

A Horse Race

Task:

1. compute $X_1, X_2, \ldots, X_n$ via $X_{t+1} = \beta + \alpha X_t + W_{t+1}$
    - $W_t \sim N(0,1)$
2. calculate and return $\frac{1}{n} \sum_{t=1}^{n} X_t$

Set $n = 10^7$

- `RBA_RBNZ_Workshops/john/fast_loop_examples`

QuantEcon

A Horse Race

Task:

1. compute $X_1, X_2, \ldots, X_n$ via $X_{t+1} = \beta + \alpha X_t + W_{t+1}$
    - $W_t \sim N(0,1)$
2. calculate and return $\frac{1}{n} \sum_{t=1}^{n} X_t$

Set $n = 10^7$

- `RBA_RBNZ_Workshops/john/fast_loop_examples`

Julia in Brief

Modern, high level, open source, scientific programming language

Strengths

- High productivity...
- and high performance!

Negatives

- Still under development
- The "rabbit hole" of advanced features (plus or minus?)

Julia in Brief

Modern, high level, open source, scientific programming language

Strengths

- High productivity...
- and high performance!

Negatives

- Still under development
- The “rabbit hole” of advanced features (plus or minus?)

Julia in Brief

Modern, high level, open source, scientific programming language

Strengths

- High productivity...
- and high performance!

Negatives

- Still under development
- The “rabbit hole” of advanced features (plus or minus?)

Why Open Source?

Lets 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 verifiable by reference to the real world. In science consensus is irrelevant. What is relevant is reproducible results. — Michael Crichton

Why Open Source?

Lets 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 verifiable by reference to the real world. In science consensus is irrelevant. What is relevant is reproducible results. — Michael Crichton

Interacting with Julia

Options

1. The REPL + a text editor (e.g., Atom or Sublime)
2. IDEs like Juno
3. Jupyter notebooks

Jupyter Notebooks

A browser based front end to Python, Julia, R, etc.

- Allows for rich text, graphics, etc.
- Easy to run remotely on servers / in cloud

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

localhost:8888/notebooks/ddp_ex_job_search_jl.ipynb

jupyter ddp_ex_job_search_jl (unsaved changes)

File Edit View Insert Cell Kernel Help Julia 0.5.0

DiscreteDP Example: Job Search

Daisuke Oyama

Faculty of Economics, University of Tokyo

We study an optimal stopping problem, in the context of job search as discussed in http://quant-econ.net/py/lake_model.html.

In [1]:
```
using QuantEcon
import QuantEcon: solve
using Distributions
using PyPlot
using Roots
```

Optimal solution

We skip the description of the model, just writing down the Bellman equation:

$$U = u(c) + \beta\,[(1-\gamma)U + \gamma E[V_s]],$$
$$V_s = \max\{U, u(w_s) + \beta\,[(1-\alpha)V_s + \alpha U]\}.$$

For this class of problem, we can characterize the solution analytically.

The optimal policy σ^* is monotone; it is characterized by a threshold s^* for which $\sigma^*(s) = 1$ if and only if $s \geq s^*$, where actions 0 and 1 represent "reject" and "accept", respectively. The threshold is defined as follows: Let

$$g(s) = u(w_s) - u(c),$$
$$h(s) = \frac{\beta\gamma}{1 - \beta(1-\alpha)} \sum_{s' \geq s} p_s u(w_s).$$

It is easy to see that g is increasing and h is decreasing. Then the threshold s^* is such that $s \geq s^*$ if and only if $g(s) > h(s)$.

Given s^*, the optimal values can be computed as follows:

$$U = \frac{\{1-(1-\alpha)\beta\}u(c) + \beta\gamma \sum_{s \geq s^*} p_s u(w_s)}{}$$

Let's try it out

- RBA_RBNZ_Workshops/john/ar1_plots_julia.ipynb