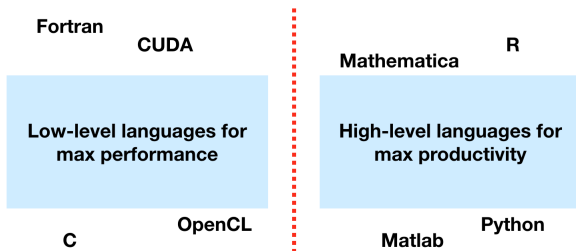


Introduction to Julia

Introduction

- ▶ Julia started in 2009
- ▶ Julia: Living in a Geekster's paradise. . .
- ▶ “Walks like Python, runs like C”
- ▶ Big idea: 2 language problem



Why I like Julia

1. It's really, really fast without special tweaks
 2. Programming syntax is close to the mathematics
 3. Parallelism is straightforward
 4. Open-source ecosystem on Github: free & tightly integrated
 5. Codebase is in Julia
 6. Can call R, Python, C, etc
 7. Speed without needing a compiler or other packages
 8. Functions do different things depending on the *types* of variables
 9. Super easy to do unit-testing
- (Based (somewhat) on Maistre (2019))

Why I don't like Julia

- ▶ Still young — was changing a lot, has settled down
- ▶ Documentation, Stack Overflow is not as good as MATLAB, Stata, R
- ▶ Not as many packages (but still a lot)
- ▶ Plotting takes a long time
- ▶ No `estout` or `stargazer`/`textables` packages yet
- ▶ Normal headaches of setting up open-source software

When I use Julia

- ▶ For computational work
 - ▶ If I have to build it from scratch
 - ▶ Dynamic programs
 - ▶ Optimization
 - ▶ Structural estimation
 - ▶ If it requires a lot of computations
- ▶ Not exploratory data analysis
- ▶ Not OLS

Getting started in Julia

- ▶ Download language: [Julia binary](#)
 - ▶ Installs to `C:\Users\username\AppData\Roaming\julia` or `~\.julia`
- ▶ Need a development environment
 - ▶ REPL is the command-line version
 - ▶ [Atom](#) is the RStudio for Julia
 - ▶ Install [uber-juno](#) package for Atom
 - ▶ [Jupyter notebooks](#) also work
 - ▶ [Microsoft VS Code](#)
 - ▶ Requires [Julia extension](#)
- ▶ Great “Getting Started” guide at [QuantEcon](#)

Adding packages & opening Jupyter

To add packages

- ▶ Open the REPL (julia command line)
- ▶ Type `]` to switch to “Package” mode
- ▶ `add IJulia`
- ▶ backspace to go back to regular prompt
- ▶ Open a Jupyter notebook

```
jupyterlab()
```

Julia scripts

Always load packages (as in R: `library(ggplot2)`)

Jupyter great for analysis using other people's functions

```
using Plots      # load Plots package
gr()             # tell it to use GR to plot (vs PyPlot, Plotly.js,

x = -10.0 : 0.5 : 10.0  # create a "range"
y = x.^2               # broadcast operation with `.^`

plot(x, y,
      title = "my first plot",
      labels = "\$x^2\$", legend = :top
)
```


Julia in Atom

Jupyter won't cut it for more serious coding. I'm developing a new version of my likelihood function.

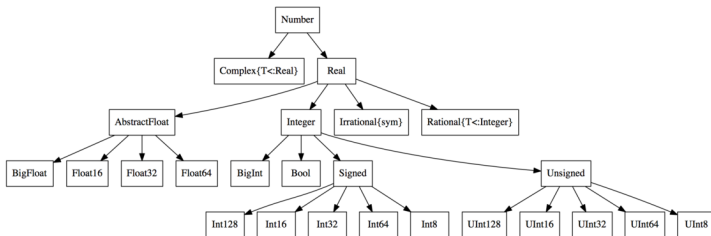
Clone package to ~/.julia/dev/ShaleDrillingLikelihood or
C:\Users\username\AppData\Roaming\julia\dev\ShaleDrillingLikelihood
with

```
] dev https://github.com/magerton/ShaleDrillingLikelihood.jl.git
```

- ▶ Project outline:
 - ▶ README, License, .gitattributes, .gitignore
 - ▶ REQUIRE file lists dependencies, superseded by new package system
 - ▶ src/ShaleDrillingLikelihood.jl defines a module (package) and loads other scripts
 - ▶ test/runtests.jl runs unit-tests
 - ▶ I build code through tests
 - ▶ Revise package reloads functions in module without restarting

Things to know about Julia

- Type system is very important



Julia resources

- ▶ Julia vs MATLAB, R Syntax
- ▶ Quantecon Lectures
- ▶ Julia Cheat Sheet
- ▶ Chapter on Julia by Jesús Fernández-Villaverde

References

Maistre, Gabriel Gauci. 2019. "10 Reasons Why You Should Learn Julia." *Medium*. Accessed September 23. <https://blog.goodaudience.com/10-reasons-why-you-should-learn-julia-d786ac29c6ca>.