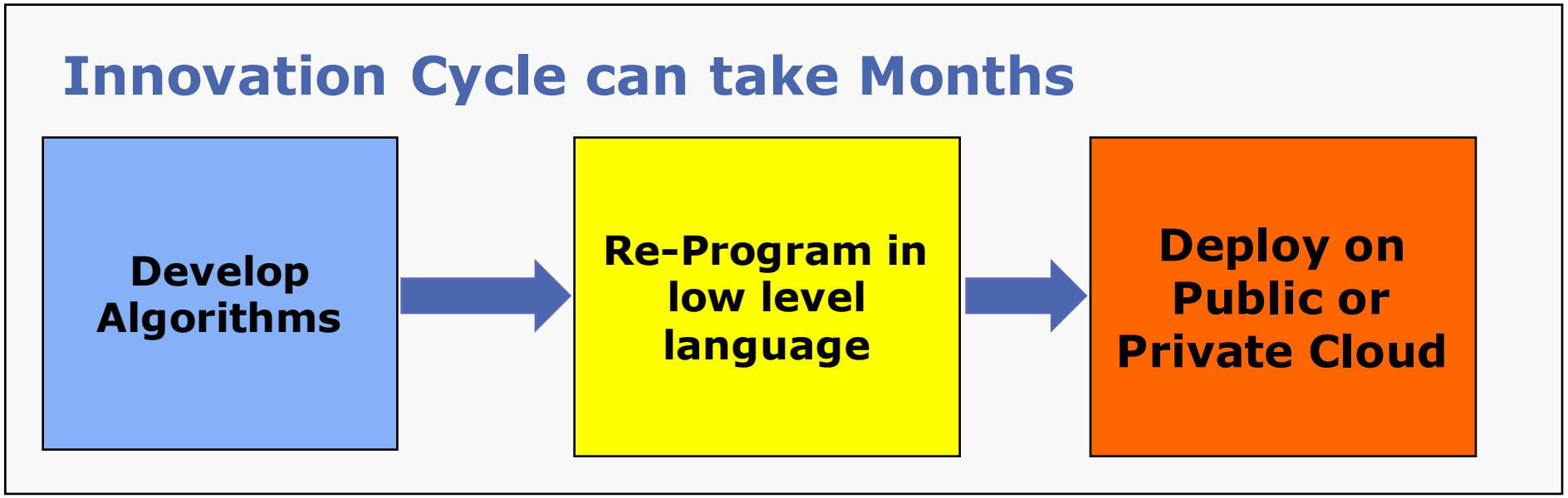
Julia in Finance

viral@juliacomputing.com

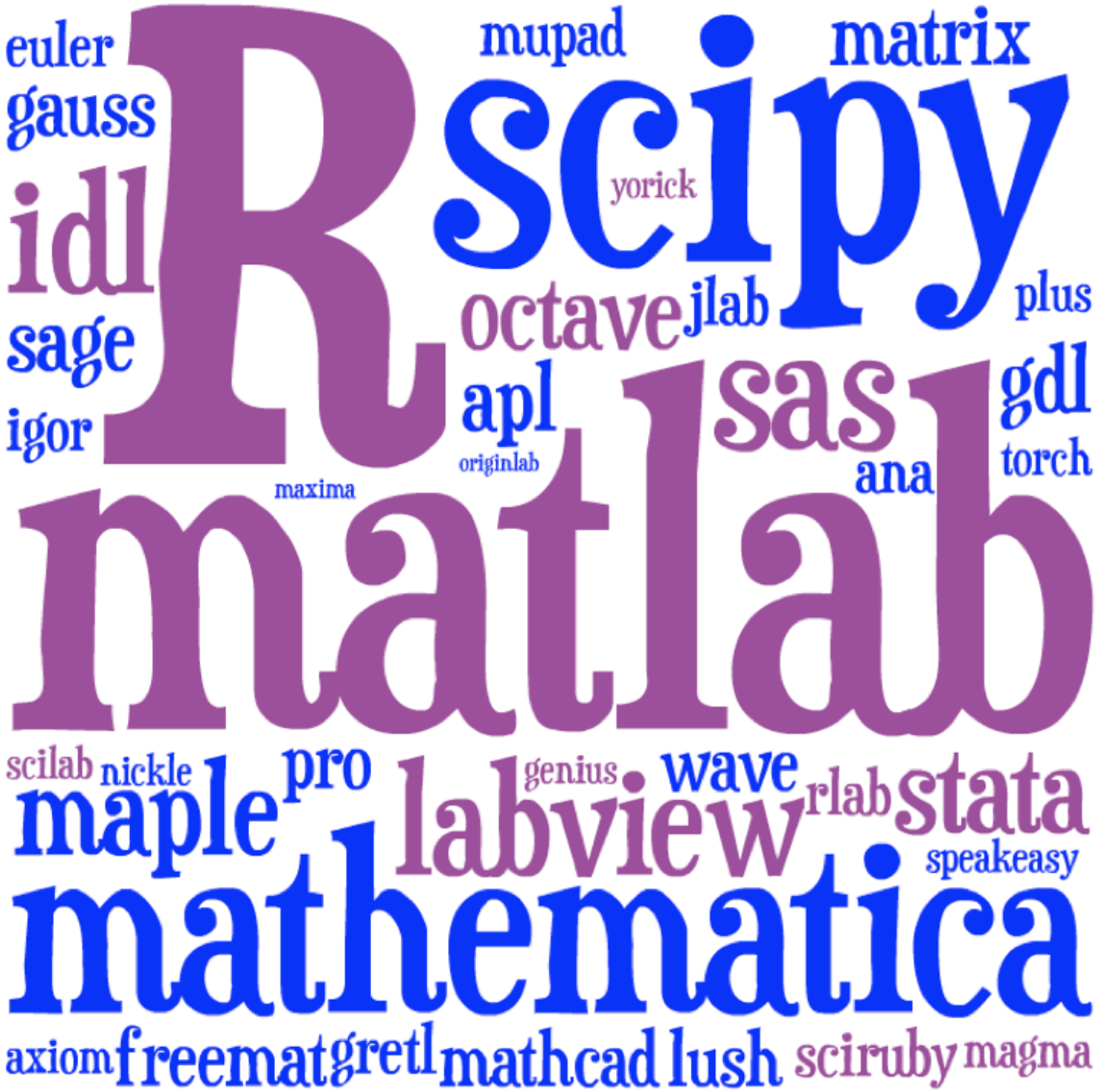
May 2016

Technology Challenges in Finance



- **Keep up with pace of business needs**
- **Compress innovation cycles**
- **Maintain competitiveness**
- **Fast evolving regulatory compliance needs**

Last 25 years…



Enter:

julia

High Productivity
High Performance
Cost-effective
Close to mathematics
Auditable code

Julia – the Strategic Choice

Strong mathematical foundation

Common language for quants, strats, engineers, data scientists and computer scientists

Julia Compresses Innovation Cycles

Today – Innovation Cycle in Months

Develop Algorithms → Re-Program in low level language → Deploy on Public or Private Cloud

Julia – Innovation Cycle in Days or Weeks

Develop Algorithms → Deploy on Public or Private Cloud

julia

The Julia edge in Finance

Julia is open source

- As productive as R, Python and Matlab
- As fast as C, Fortran and Java.

Julia Professional Edition

- IDE with
  - Compiler
  - Debugger
  - Profiler
- Plotting libraries
- Math libraries

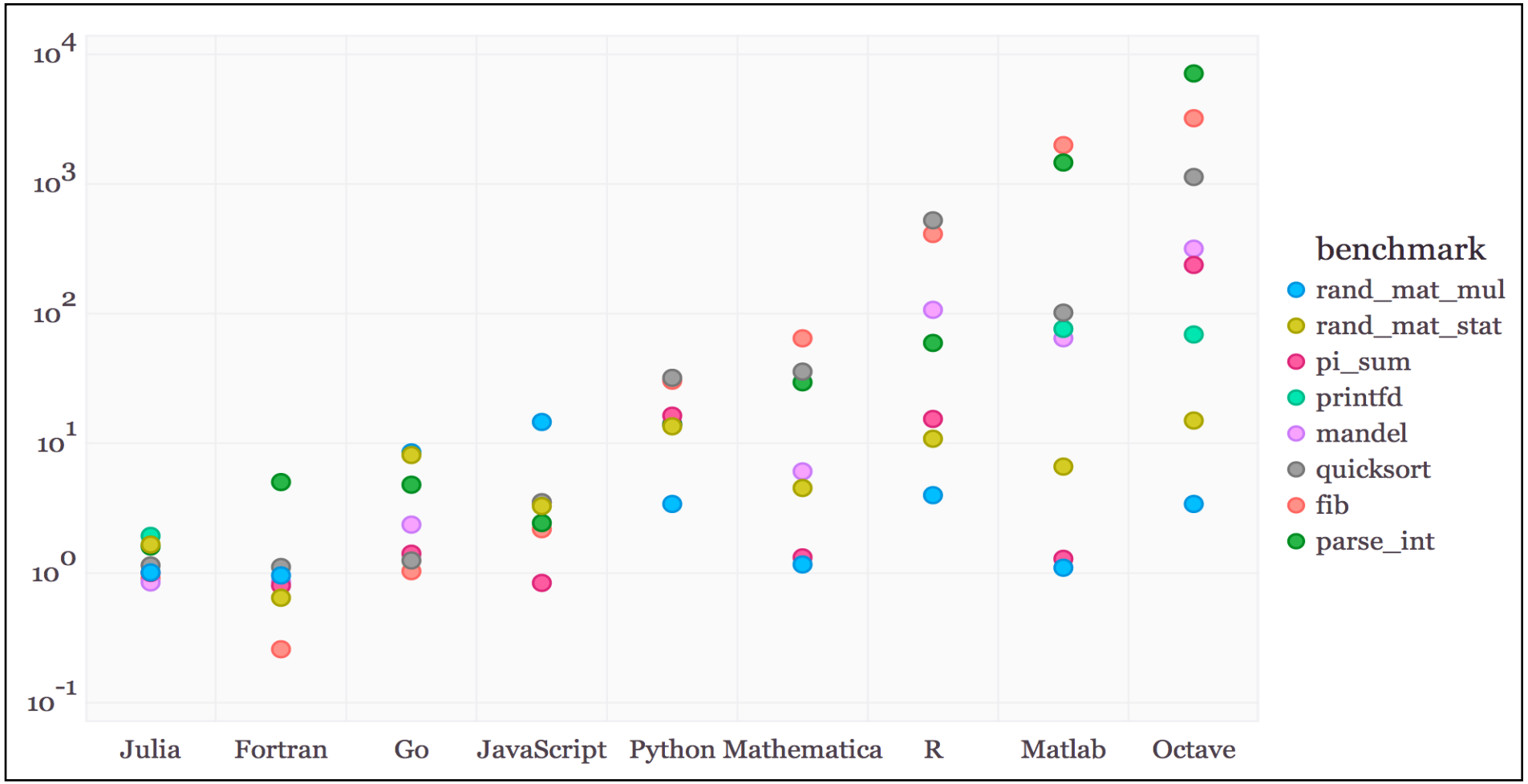
Julia Deployment Edition

- Single click deployment
- Horizontal scalability
- Monitoring

Integration with finance software

- Excel
- Bloomberg
- Databases
  - Oracle
  - KDB
  - MySQL
  - …

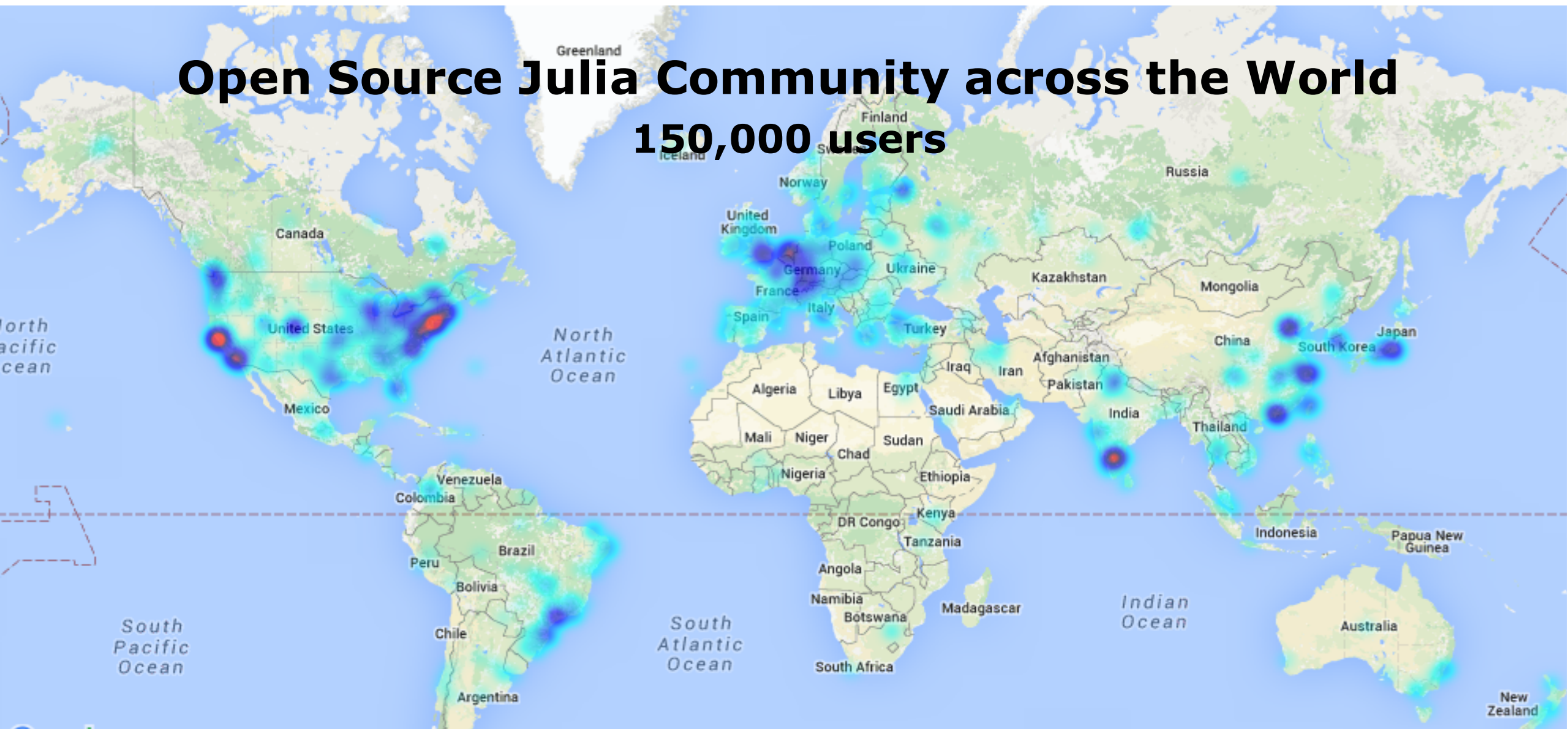
Julia Performs as well as C and Fortran



Performance benchmark relative to C. A value of 1 means as fast as C. Lower values are better.

Open Source Julia Community across the World

150,000 users



Nerve Centres in Boston, New York, and Bangalore

Research anchored at MIT

Julia in Finance

- Customers of Julia Computing include **3 of the top 10 banks**
- Use cases include:
  - Asset allocation
  - Asset pricing
  - Portfolio optimization
  - Risk computations
  - Trading systems
- Solutions developed by quants and strats in Julia can be deployed in Julia – **saving $$$** in rewriting and faster time to market
- Julia code is much more readable, hence ideal for any solution where **regulatory inspection** is necessary

A small example: Black Scholes option pricing in Julia

```julia
# Compute Black Scholes

cndf2(in::Vector{Float64}) = 0.5 .+ 0.5 .* erf(0.707106781 .* in)

function blackscholes(sptprice::Vector, strike::Vector,
                      rate::Vector, σ::Vector, time::Vector)

    logterm = log10(sptprice ./ strike)
    powterm = 0.5 .* σ .* σ
    den = σ .* √time
    δ₁ = (((rate .+ powterm) .* time) .+ logterm) ./ den
    δ₂ = δ₁ .- den
    NofXd1 = cndf2(d1)
    NofXd2 = cndf2(d2)
    futureValue = strike .* exp(-rate .* time)
    c1 = futureValue .* NofXd2
    call = sptprice .* NofXd1 .- c1
    put  = call .- futureValue .+ sptprice

end
```

- As easy as a scripting language
- As fast as C
- Conciseness enables auditability
- Unicode variable names allow code to mirror the mathematics
- Large mathematical library

Julia for Fundamental Review of Trading Book – FRTB Regulation

- Large scale changes to bank capital requirements, due to go live early 2018
- Much greater complexity, and increased computational load
  - Many more risk factors
  - Expected Shortfall needs more scenarios for useable results
  - Continuous monitoring of internal models vs standard approach
  - Varying liquidity horizons
- Handling all this additional complexity in C / C++ is daunting
  - Programmer productivity – get this done in the next two years?
  - Will the new models be traceable to management / regulator
- Significant increase in computational capacity required
  - Dynamic languages might be too slow for large trading books and / or near real time analyses

Julia for Solvency II Regulation

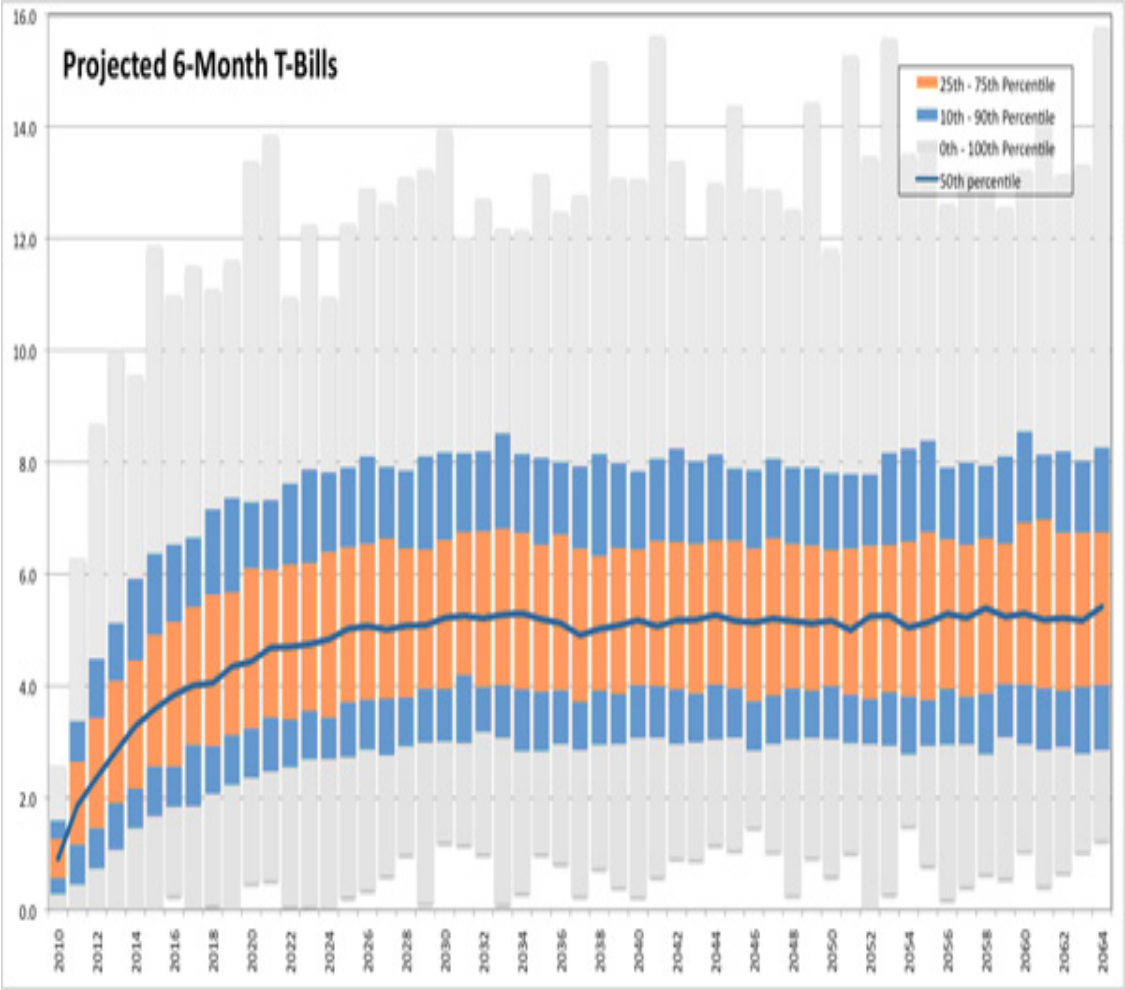
- Solvency II regulation significantly increases quantitative requirements, risk management and transparency
- Insurers are required to compute risk sensitive capital numbers and hold adequate capital to cover them
- Much larger computations, quantitative assessment and simulations on a daily basis
- Julia is ideally suited with high productivity and high performance

Case Study 1: Julia at a Top-10 Insurer

- New European Solvency II capital regime
- Replacing old system that uses IBM Algorithmics
- Faster:
  - Julia model is **20x faster** than comparable R code and **up to 1000x faster** than Algorithmics
  - Evaluates millions of scenarios
- Better:
  - Julia solution is in-house (1000 Lines of Code as against 14,000 Lines of Code with Algorithmics)
  - Elegant, readable, inspect'able code
  - Cost-effective

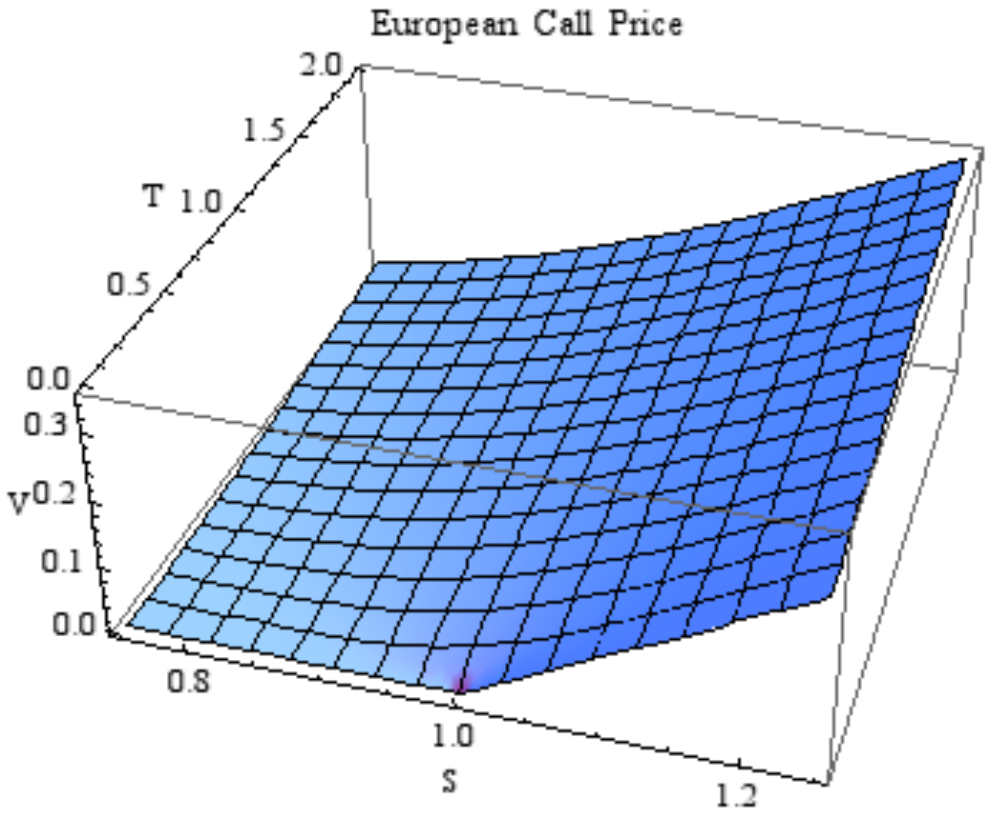
Case Study 2: Asset Management Company

- An economic scenario generator
- High-dimensional data set on which data extraction, data reordering, and various statistical kernel computations are performed
- Faster:
  - Original code was in K
  - Julia code is 4x-10x faster
- Better:
  - Julia code is significantly more readable
  - Easy to maintain and update
  - Cost-effective



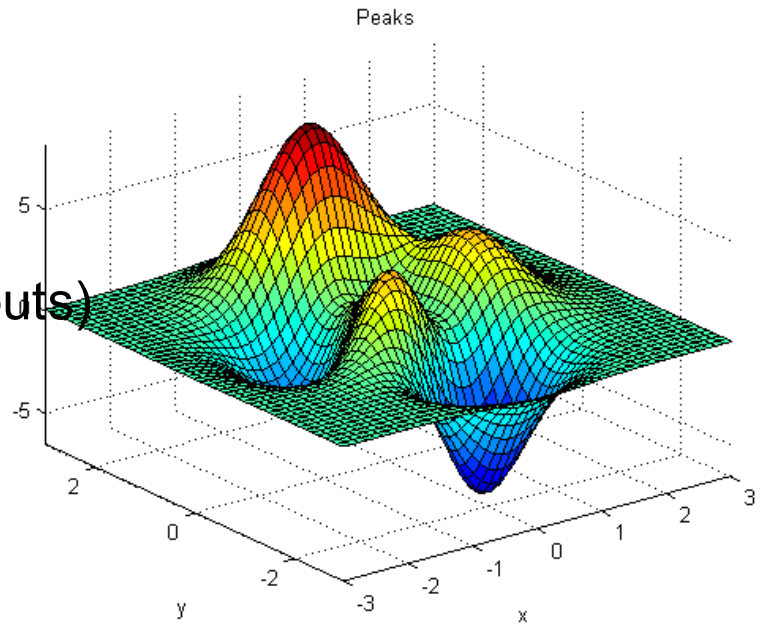
Case Study 3: Industry Standard Derivatives Benchmark STAC-A2

- Industry standard benchmark for computational finance and risk analysis
- Monte-Carlo pricing of European and American options, as well as various Greeks computation
- Various kernels
  - Random number generation
  - Asset path generation using Andersen QE method
  - Early-exercise via Long-Staff Schwartz method (SVD based)
- Productivity
  - Lines of Code on par with reference R implementation
- Performance
  - Performance within 2x of ninja C code produced by Intel
  - Reference R code unable to run on the full specification



Case Study 4: Optimization at a Top-10 Fund Manager

- **Solving a large complex mathematical optimization problem for mortgages**
- **Full optimization: (Faster Speed + Better Quality)**
  - MATLAB 2014a 558.094600 seconds, 3110 iterations
  - Julia v0.4 **1.833 seconds, 50 iterations (300x faster)**



- **Performance: Objective function only** (100 iterations with same inputs)
  - MATLAB 2014a 2.69 seconds
  - Julia v0.4 **0.78 seconds (3.5x faster)**
- **Quality**: Optimization value (11-parameter)
  - MATLAB 2014a 4.277644613116166e+14 (3110 iterations)
  - Julia v0.4 **4.270887086707642e+14 (50 iterations)**

Case Study 5: Technical Analysis with Julia

AlgoCircle uses Julia for automating technical analysis:

- **Time series data processing**
- **Implementation of Technical Indicators**
- **ML using decision trees for trade signals**
- **Charting**

Case Study 6: Economic Models at the NY Fed

FEDERAL RESERVE BANK *of* NEW YORK *Serving the Second District and the Nation*

About the New York Fed | Markets & Policy Implementation | Economic Research | Financial Institution Supervision

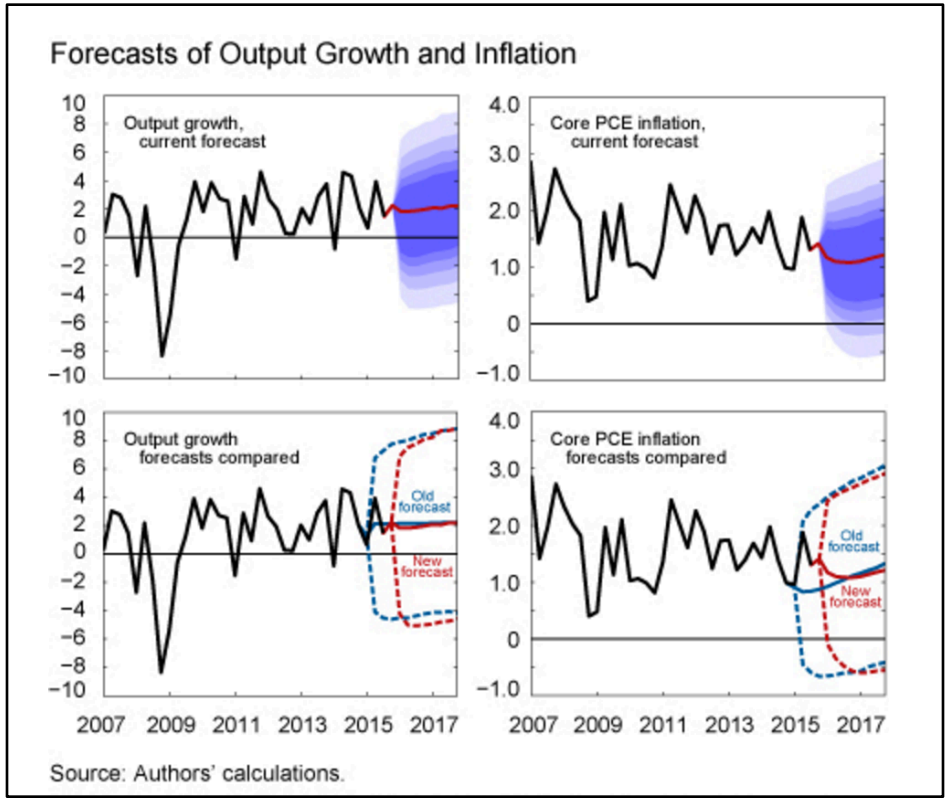
Liberty Street Economics

« Just Released: Job Market Remains Tight as Regional Economy Slows | Main | At the New York Fed: Conference on the Evolving Structure of the U.S. Treasury Market »

DECEMBER 03, 2015

The FRBNY DSGE Model Meets Julia

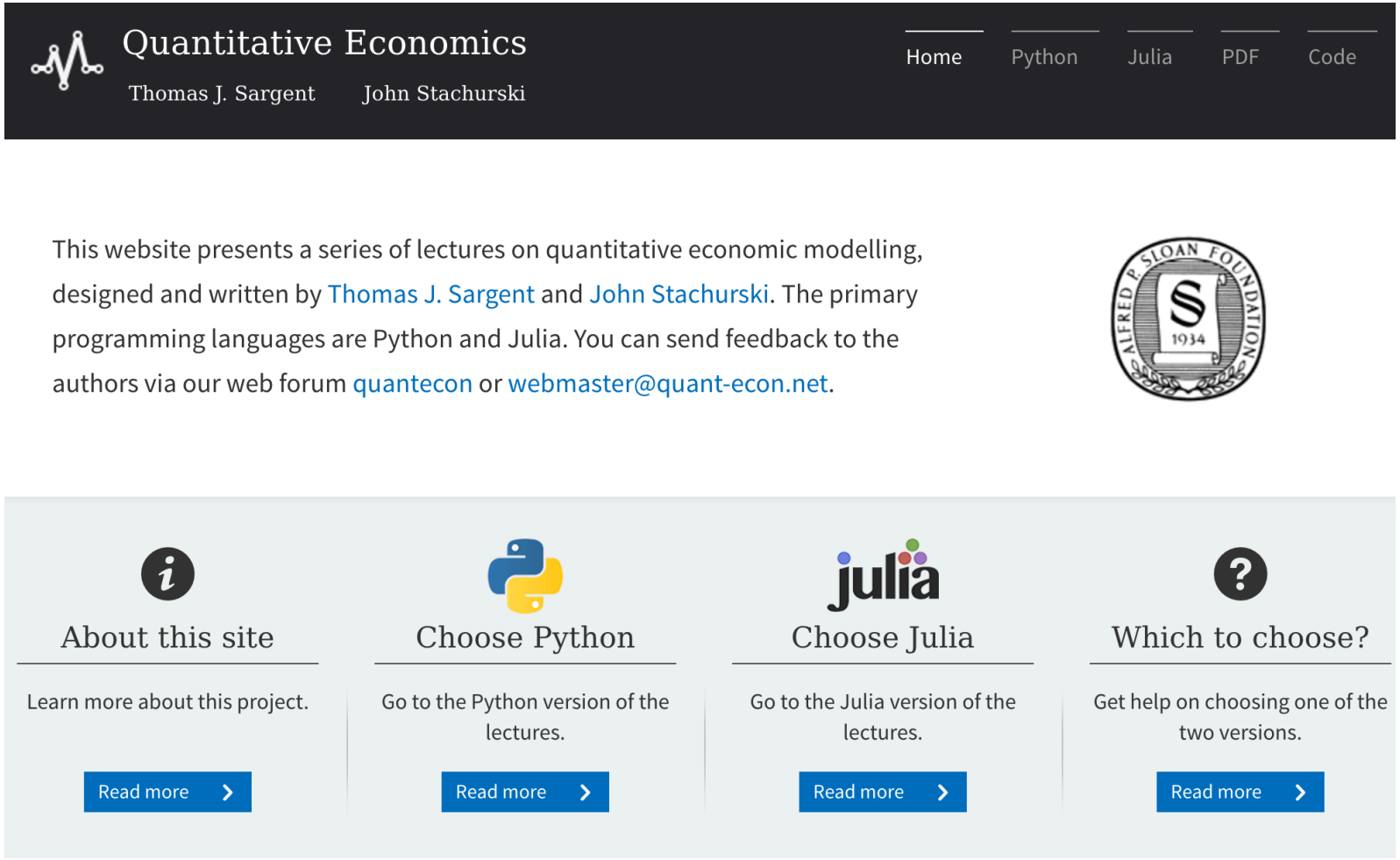
Marco Del Negro, Marc Giannoni, Pearl Li, Erica Moszkowski, and Micah Smith



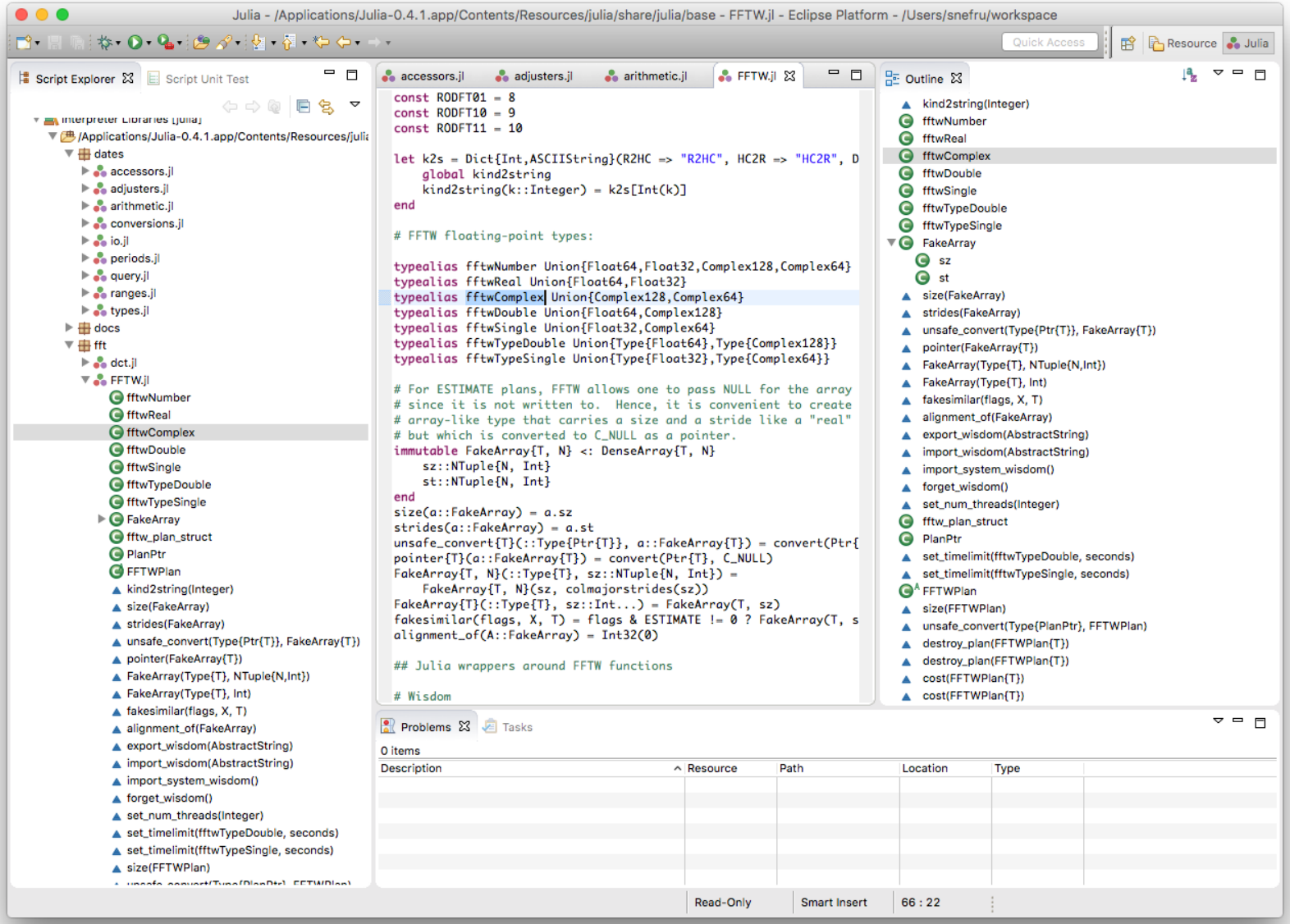
"We tested our code and found that the model estimation is about ten times faster with Julia than before, a very large improvement. Our ports of certain algorithms, such as Chris Sims's gensys (which computes the model solution), also ran about six times faster in Julia".

Source: http://libertystreeteconomics.newyorkfed.org/2015/12/the-frbny-dsge-model-meets-julia.html

Case Study 7: Graduate Textbook on Macroeconomics by Nobel Laureate Thomas Sargent



Julia Professional – Based on Eclipse



JuliaBox beta

Run Julia from the Browser. No setup.

The Julia community is doing amazing things.

We want you in on it!

Sign in via Google

IJulia

Create IJulia Notebooks and share them.

Console

Use in-browser terminal emulator to fully control your Docker instance.

Google Drive

Collaborate with others. Sync notebooks and data via Google Drive.

Sync & Share

Setup folders to sync with remote git repositories.

Julia-Excel Integration

IE9 - Win7 (Clean install Snapshot) [Running]

Juno/LT

File Edit View Evaluation Help

Tutorial.jl* arb.jl

```julia
  r = fzero(r->apply_discount(r, Pc, F, C, n, m), [0, 1000])
end

function apply_discount(r, Pc, F, C, n, m)
    C_discounted = 0.0
    u = (1 + r)^(1.0/n)
    F_discounted = F / u^m
    for t=1:m+1
        C_discounted += C / u^t
    end

    return Pc - (C_discounted + F_discounted)
end
```

Main 38 / 32

```
julia
  _   _ _(_)_     |  Version 0.4.2 (2015-12-06 21:47 UTC)
 (_) | (_) (_)    |  Official http://julialang.org/ release
 _ _ _| |_  __ _   |  i686-w64-mingw32
julia> include("C:\\Users\\IEUser\\Documents\\arb.jl")
WARNING: Method definition info(Any...) in module Base at util.jl:334 overwritten in module Logging at C:\Users\IEUser\.julia\v0.4\Logging\src\Logging.jl:61.
WARNING: Method definition warn(Any...) in module Base at util.jl:364 overwritten in module Logging at C:\Users\IEUser\.julia\v0.4\Logging\src\Logging.jl:61.
3/6/2009
Task (queued) @0x050ed8d03/6/2009

julia>
julia>

julia> 3/6/2009
3/6/2009
3/6/2009
julia>

julia>

julia>
```

Securities [Compatibility Mode] - Excel

File Home Insert Page Layout Formulas Data Review View Tell me Sign in Share

SUM =jlcall("compute_yield", B7, H7, G8/4, 4, J7)

We get the prices of various Preferred securities of Aegon on March 6, 2009, in the middle of the financial cri
Based on the market prices prevailing on that day, we compute the implied interest rate of all the various sec
Given that all these are of the same company, a difference in the implied interest rate is an arbitrage opportu

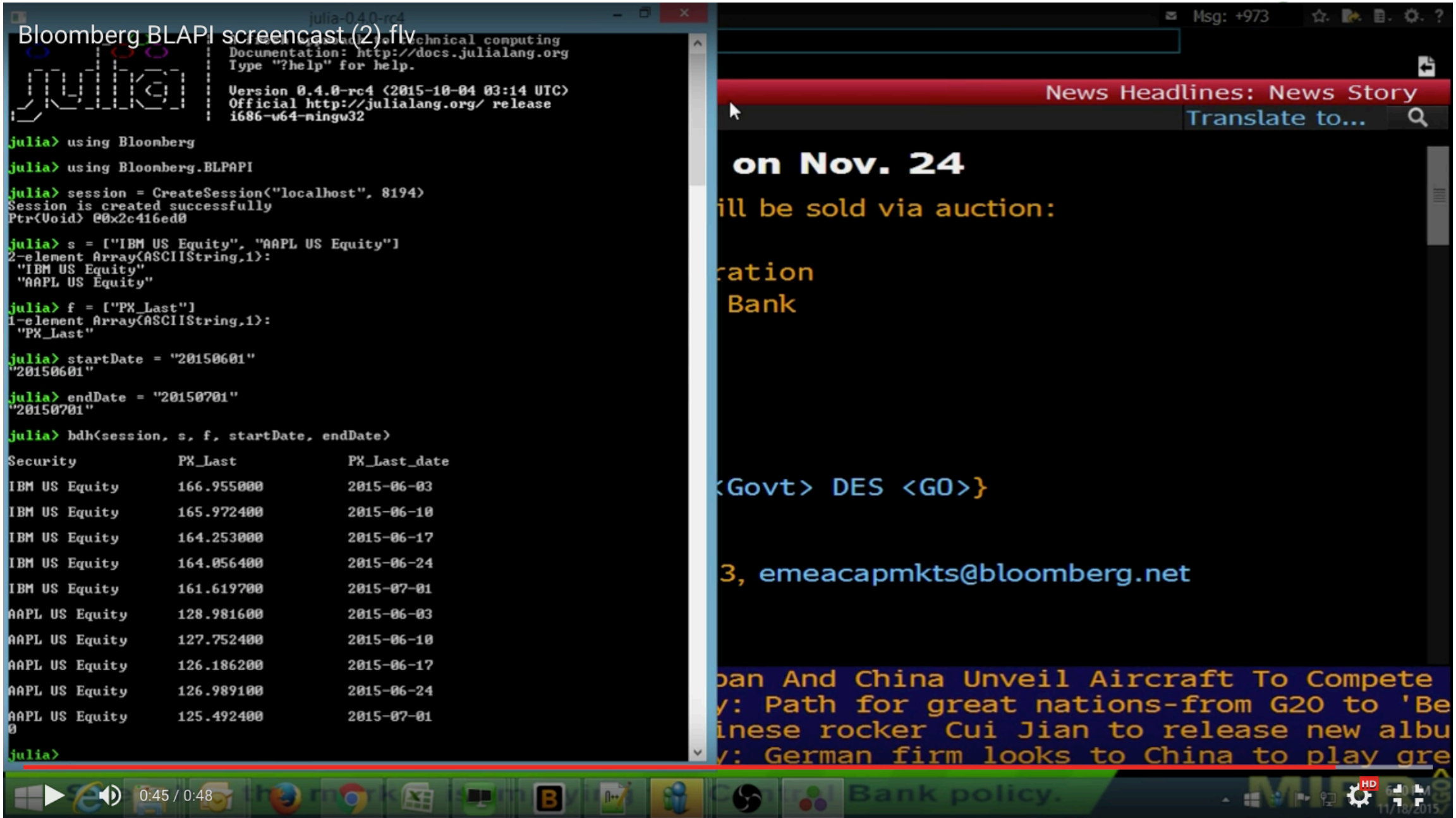
Symbol	Price	Interest	Security Description	Stock Exchang	IPO Date	Cpn Rate	LiqPref	Call Date	Coupons to Maturity	Moodys S&P
CUSIP	4/6/2010					Ann Amt	CallPrice	Matur		Dated
AEH	19.97	=jlcall("compu	AEGON N.V., 6.375% Perpetual Capital	NYSE		6.38%	$25.00	6/15/2015	23	Baal / A
7924301				Chart	5/26/2005	$1.59	$25.00	None		3/3/2009
AED	20.05	46.61%	AEGON N.V., 6.50% Perpetual Capital Securities	NYSE		6.50%	$25.00	12/15/2010	3	Baal / A
7924400				Chart	11/18/2005	$1.63	$25.00	None		3/3/2009
AEV	21.23	19.17%	AEGON N.V., 6.875% Perpetual Capital	NYSE		6.88%	$25.00	9/15/2011	7	Baal / A
N00927306				Chart	6/22/2006	$1.72	$25.00	None		3/3/2009
AEF	22.45	12.79%	AEGON N.V., 7.25% Perpetual Capital Securities	NYSE		7.25%	$25.00	12/15/2012	11	Baal / A
N00927348				Chart	9/17/2007	$1.81	$25.00	None		3/3/2009
AEB	20.35	invalid data : 3	N.V., Floating Rate Perpetual Capital	NYSE		Floating	$25.00	12/15/2010	3	Baal / A
7924509				Chart	11/18/2005	n.a.	$25.00	None		3/3/2009

Master Sheet AEGON

Edit 90%

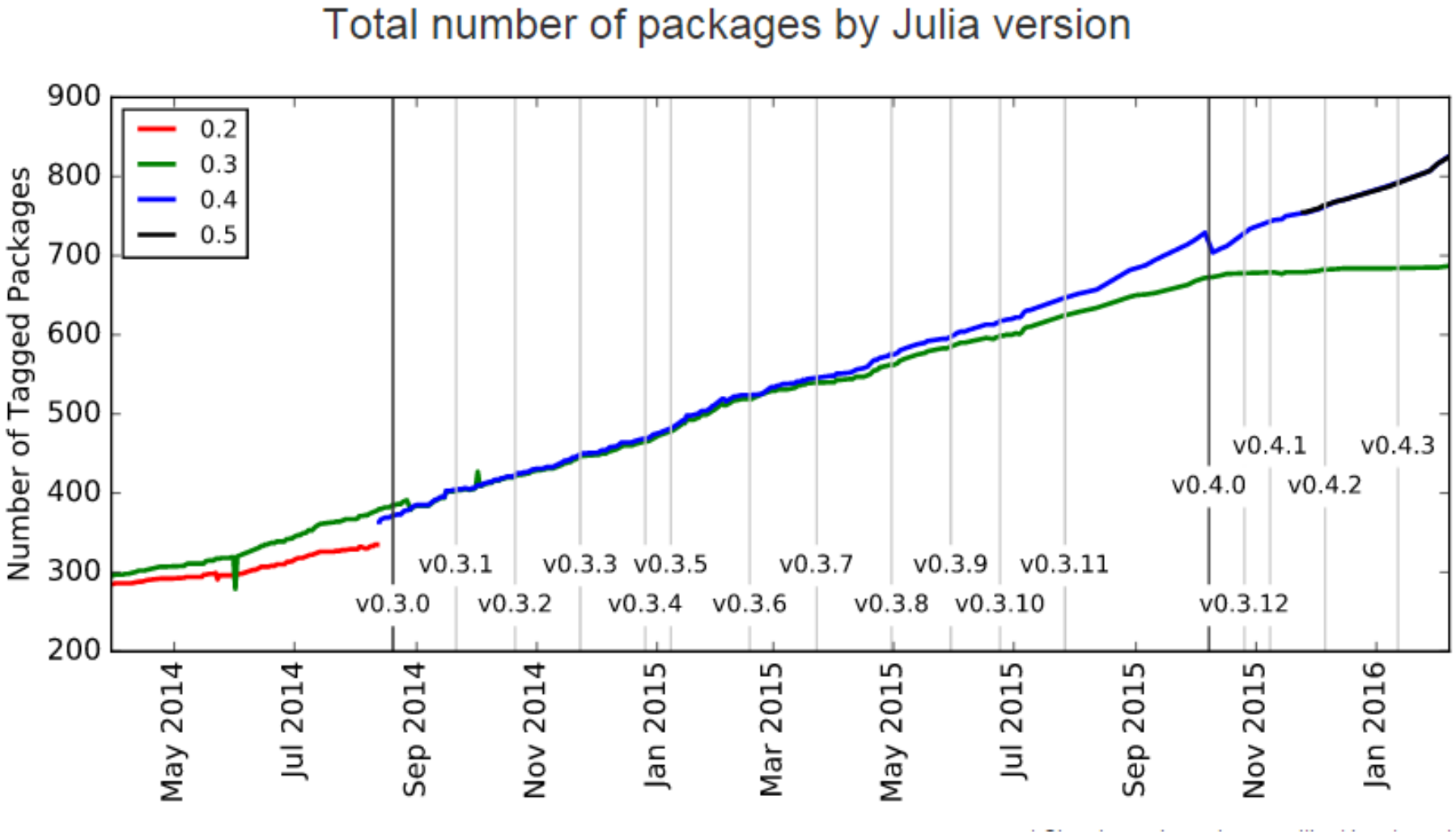
11:11 AM 12/26/2015

Julia-Bloomberg Integration



Julia Package Ecosystem Pulse

Last updated 2016-02-07



http://pkg.julialang.org/pulse.html

High Calibre Founding Team

Committed to the Long Term Success of Julia

Prof. Alan Edelman

- Professor, Applied Math and CS at MIT
- Head of Julia Research Team at MIT since 2009
- Founder of Interactive Supercomputing (Acquired by Microsoft)
- Co-creator, Star-P (parallel MATLAB)
- Co-Creator of Julia
- Various awards: http://en.wikipedia.org/wiki/Alan_Edelman#Awards

Dr. Viral Shah

- Co-Creator & core contributor of Julia
- Co-Creator of Circuitscape and Star-P
- Co-Founder, FourthLion Technologies
- Architect of subsidy reforms, Govt. of India
- Co-author of the bestseller book: Rebooting India
- PhD, Computational Sciences, UC Santa Barbara

High Calibre Founding Team

Committed to the Long Term Success of Julia

Deepak Vinchhi

- Successful Serial Entrepreneur; 20+ years in enterprise software
- Founder & CEO of Aquila Technologies (Acquired by HCL Tech)
- Managing Director, Sterling Commerce India; Sterling was acquired by IBM; Headed $1 B. Industry Solutions Products Portfolio at IBM
- BTech, IIT-Bombay; MS, University of Cincinnati

Dr. Jeff Bezanson

- Co-Creator & core contributor of Julia
- Designed the M# Matlab compiler at ISC
- BA Computer Science, Harvard
- MS Computer Science, MIT
- PhD, MIT

High Calibre Founding Team

Committed to the Long Term Success of Julia

Stefan Karpinski

- Co-Creator & core contributor of Julia
- Data Scientist at Etsy
- Worked at Akamai and Citrix Online
- BA Mathematics, Harvard

Keno Fisher

- Core contributor to Julia
- Author of key functionality: Windows, debugger, etc.
- Started hacking on Julia in high school
- Currently at Harvard; expected MA in Physics in 2016

Julia computing

How can we help you?

viral@juliacomputing.com