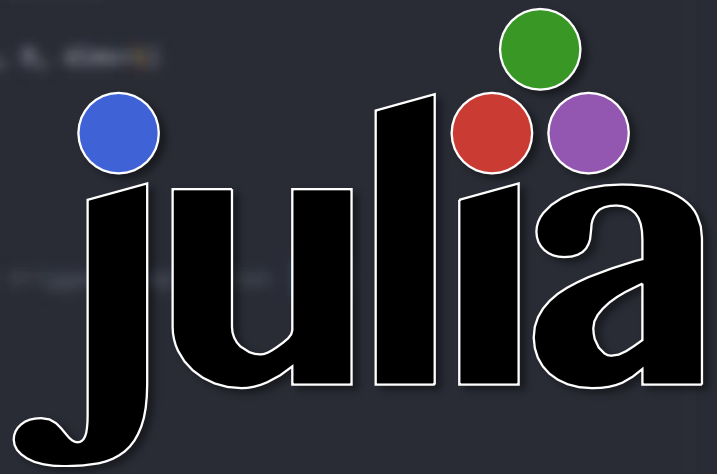


Advanced Julia Workshop

Carsten Bauer @ University of Cologne, October 2019

thp Institute for Theoretical Physics University of Cologne

bcgs Bonn-Cologne Graduate School of Physics and Astronomy

https://science.raphael.poss.name/programming-levels.html

How good are you at programming?
A CEFR-like approach to measure programming proficiency

Raphael 'kena' Poss

A1 level → Workshop → B1 level

Programming skills self-assessment matrix

		A1 Basic User	A2 Basic User	B1 Intermediate User	B2 Intermediate User	C1 Proficient User	C2 Proficient User
Writing	Writing code	I can produce a correct implementation for a simple function, given a well-defined specification of desired behavior and interface, without help from others.	I can determine a suitable interface and produce a correct implementation, given a loose specification for a simple function, without help from others. I can break down a complex function specification in smaller functions.	I can estimate the space and time costs of my code during execution. I can empirically compare different implementations of the same function specification using well-defined metrics, including execution time and memory footprint. I express invariants in my code using preconditions, assertions and post-conditions. I use stubs to gain flexibility on implementation order.	I use typing and interfaces deliberately and productively to structure and plan ahead my coding activity. I can design and implement entire programs myself given well-defined specifications on external input and output. I systematically attempt to generalize functions to increase their reusability.	I can systematically recognize inconsistent or conflicting requirements in specifications. I can break down a complex program architecture in smaller components that can be implemented separately, including by other people. I can us existing (E)DSLs or metaprogramming patterns to increase my productivity.	I can reliably recognize when under-specification is intentional or not. I can exploit under-specification to increase my productivity in non-trivial ways. I can devise new (E)DSLs or create new metaprogramming patterns to increase my productivity and that of other programmers.
	Refactoring	I can adapt my code when I receive small changes in its specification without rewriting it entirely, provided I know the change is incremental. I can change my own code given detailed instructions from a more experienced programmer.	I can determine myself whether a small change in specification is incremental or requires a large refactoring. I can change my own code given loose instructions from a more experienced programmer.	I can derive a refactoring strategy on my own code, given relatively small changes in specifications. I can change other people's code given precise instructions from a person already familiar with the code.	I can predict accurately the effort needed to adapt my own code base to a new specification. I can follow an existing refactoring strategy on someone else's code. I can take full responsibility for the integration of someone else's patch onto my own code.	I can reverse-engineer someone else's code base with help from the original specification, and predict accurately the effort needed to adapt it to a new specification.	I can reverse-engineer someone else's code base without original specification, and predict accurately the effort needed to adapt it to a new specification.

Workshop Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9:30 - 11:30 (2h)	Types + Dispatch	Environments + Julia Gotchas	Parallel Computing	**Holiday**	**Hackathon** + MiniWorkshops
	Short break				
11:45 - 12:45 (1h)	Custom types	Linear Algebra + Quantum Ising	Distributed		
	Lunch break				
14:00 - 16:00 (2h)	Specialization + Generic Prog.	Machine Learning Physics	Tips&Tricks + Ecosystem		


Microsoft
Reactor London
juliacon
2018

Workshop materials

Terminal

```
git clone https://github.com/crstnbr/julia-workshop-2019

cd julia-workshop-2019

julia install_dependencies.jl
```

Workshop materials

Julia REPL

```
using Pkg
Pkg.GitTools.clone("http://tiny.cc/julia", "workshop")
cd("workshop")
include("install_dependencies.jl") # might take a while
```

Workshop materials

Browser

Download `https://github.com/crstnbr/julia-workshop-2019/archive/master.zip`

Unzip the file.

Open the Jupyter notebook `install_dependencies.ipynb` and execute the cells.

What does science need from a programming language?

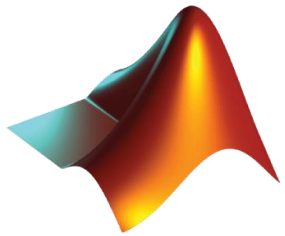
Easy to write and read !

Fast and scalable !

Interactive !

There’s a plethora of programming languages

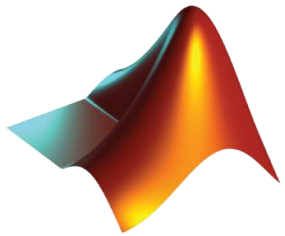
There’s a plethora of programming languages



There’s a plethora of programming languages

Fast

Slow(ish)

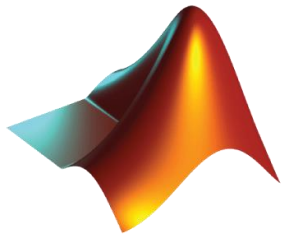


MATLAB

There’s a plethora of programming languages

Compiled

Interpreted



MATLAB

There’s a plethora of programming languages

Static

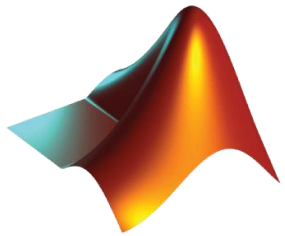
Dynamic

There’s a plethora of programming languages

Fortran

Speed

Convenience



MATLAB

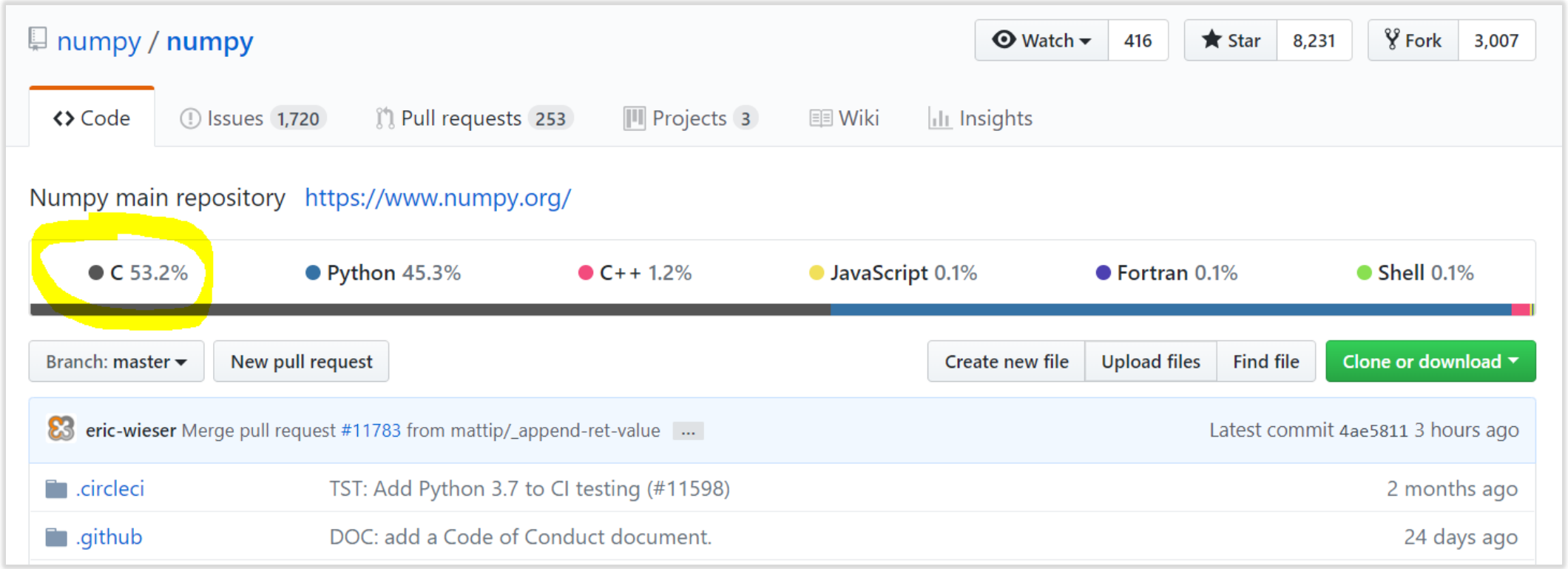
The “two language problem”

a.k.a Ousterhout's dichotomy

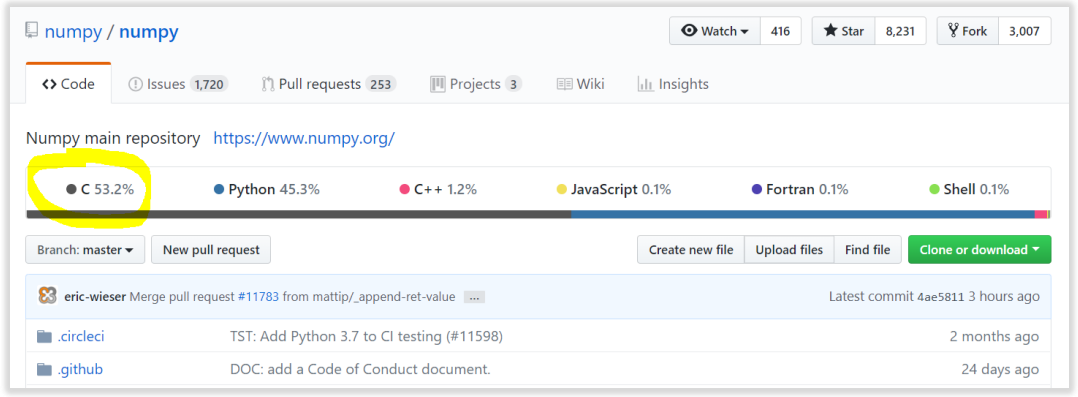
Prototype

Production

The “two language problem”

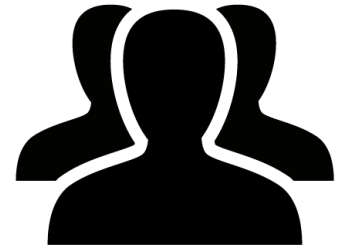


The “two language problem”



Developer

User



The “two language problem”

static	dynamic
compiled	interpreted
user types	standard types
standalone	glue

dynamic

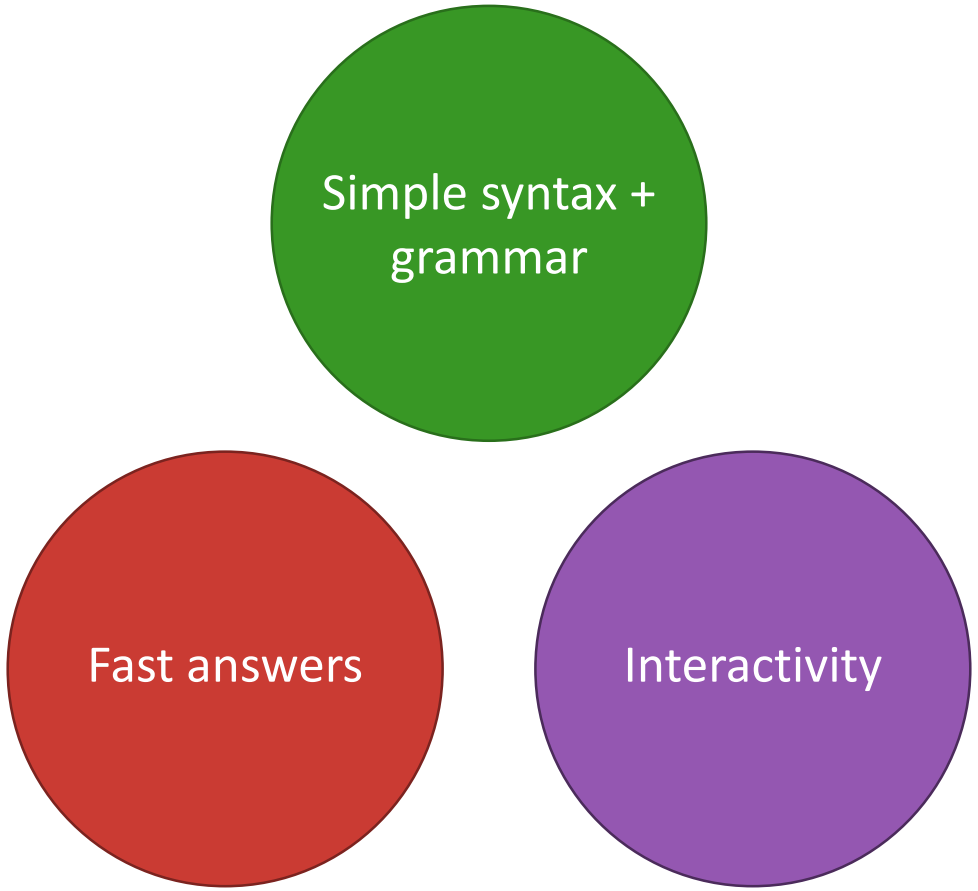
compiled

user types **and** standard types

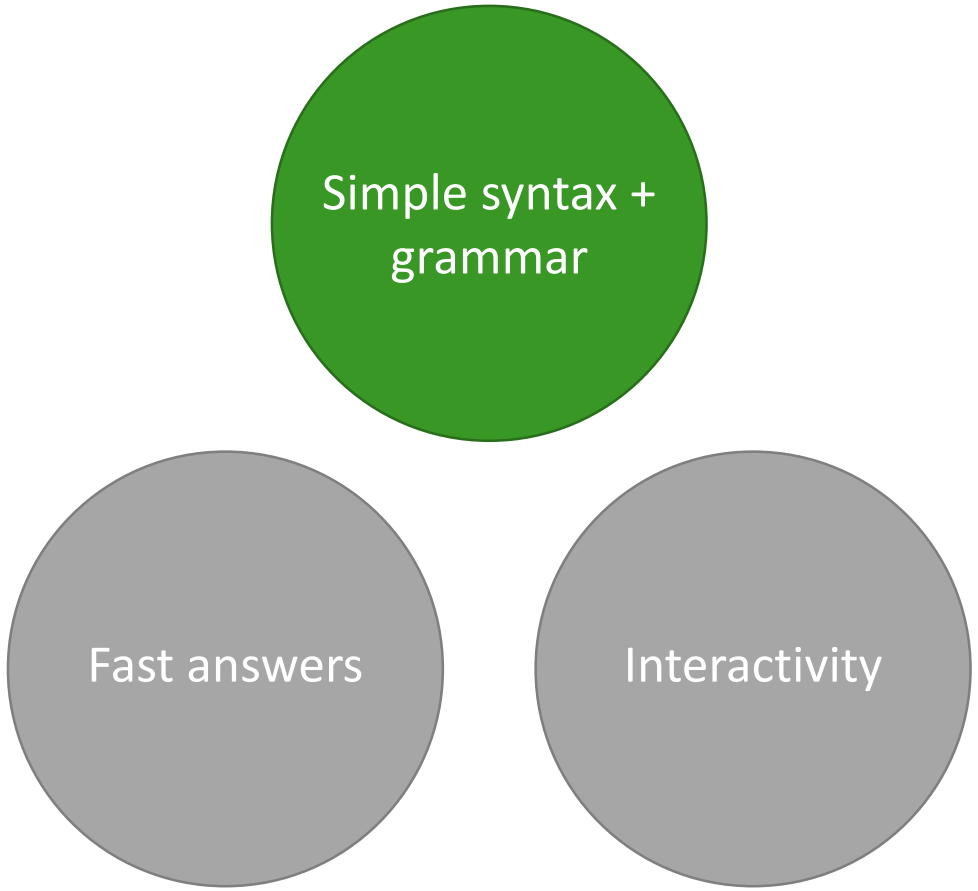
standalone **or** glue

“Feels like Python, runs like C”

The julia unification



The julia unification



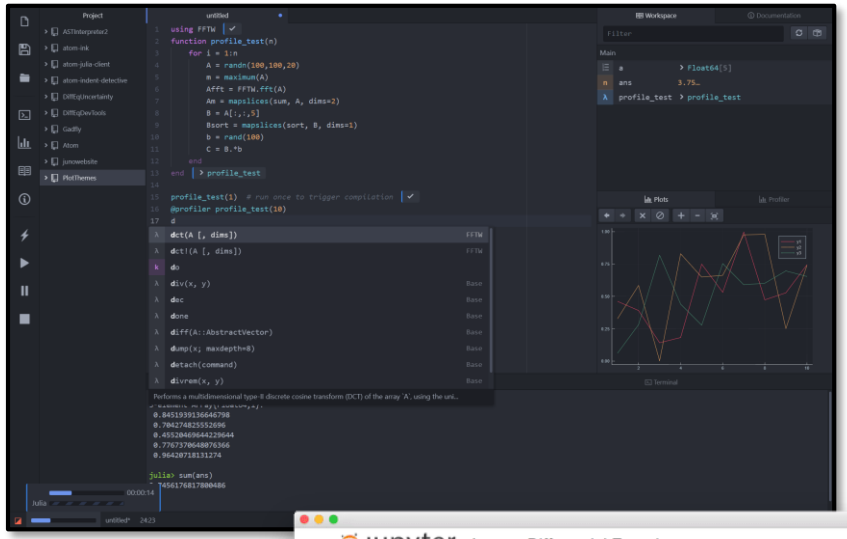

```
function babylonian(α; N = 10)
    @assert α > 0 "α must be > 0"
    t = (1+α)/2
    for i = 2:N
        t = (t + α/t)/2
    end
    t
end

babylonian(π) ≈ √π
```

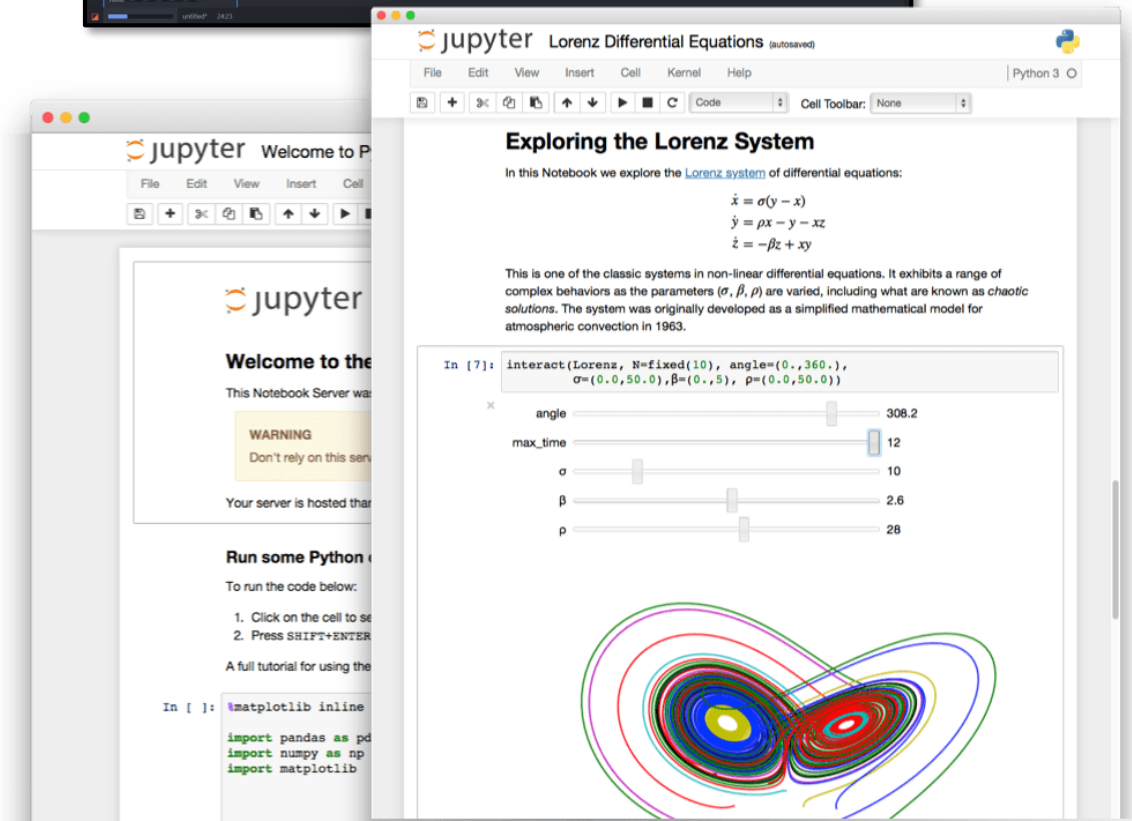
The julia unification

Simple syntax + grammar

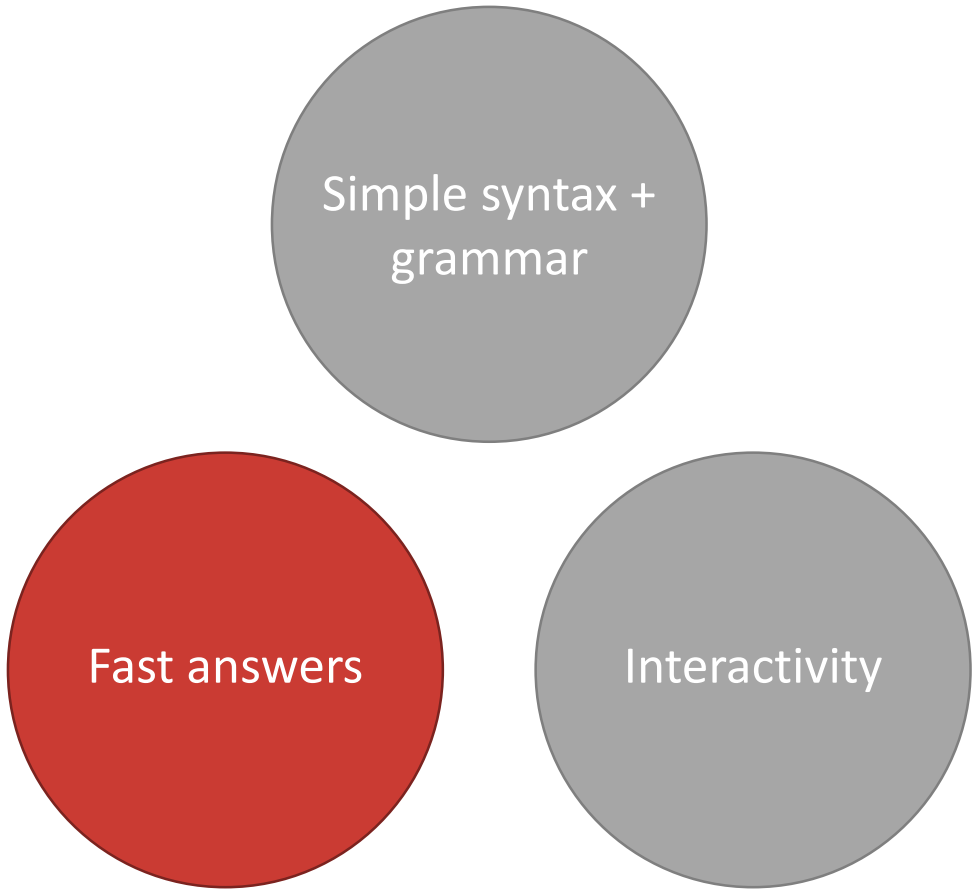
Fast answers

Interactivity





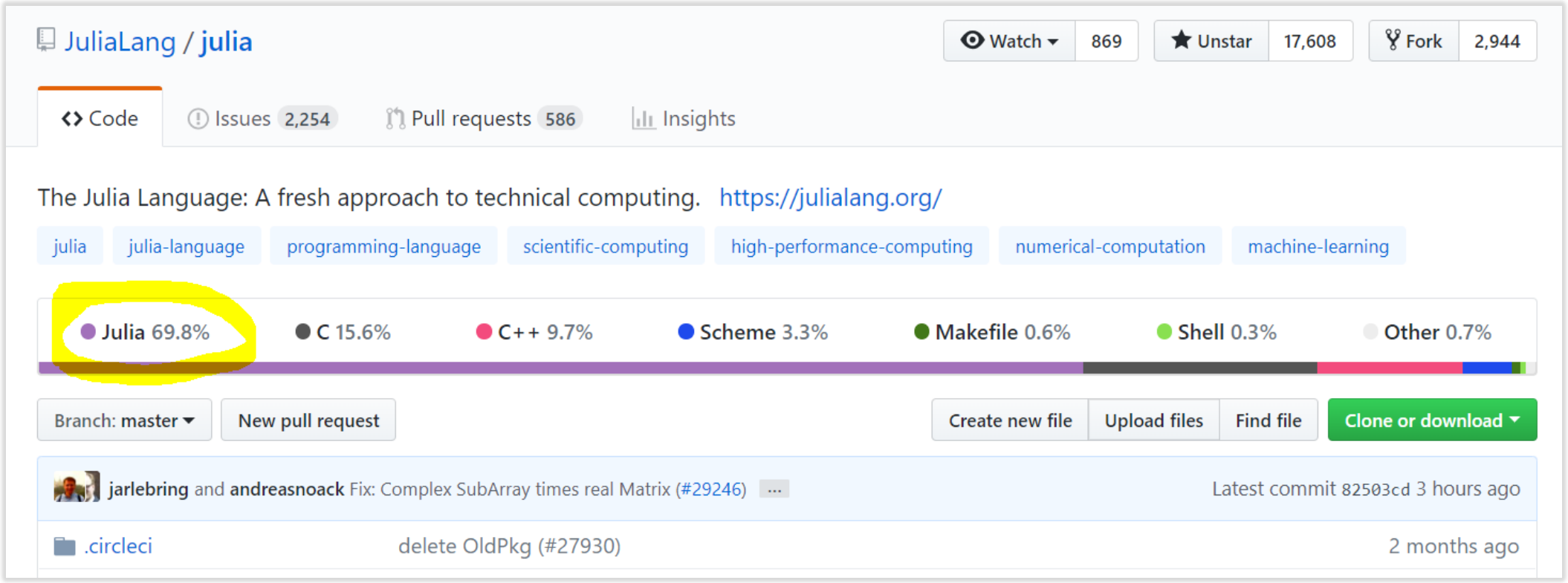
The julia unification



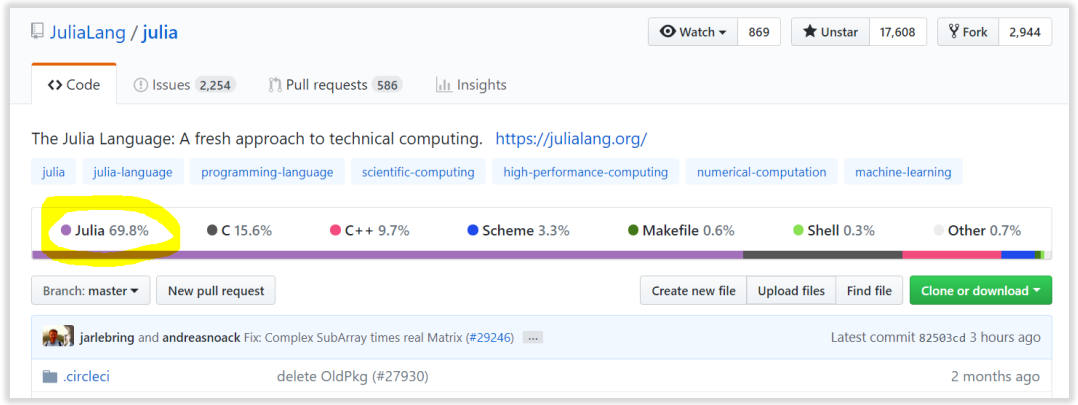

```
julia> function sumup()
           x = 0
           for i in 1:100
               x += i
           end
           x
       end
sumup (generic function with 2 methods)

julia> @code_llvm debuginfo=:none sumup()

; Function Attrs: uwtable
define i64 @julia_sumup_12626() #0 {
top:
  ret i64 5050
}
```

Just returns the answer!
(The for loop was compiled away)

It is free and open source

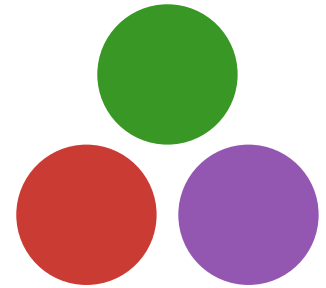


It is inviting

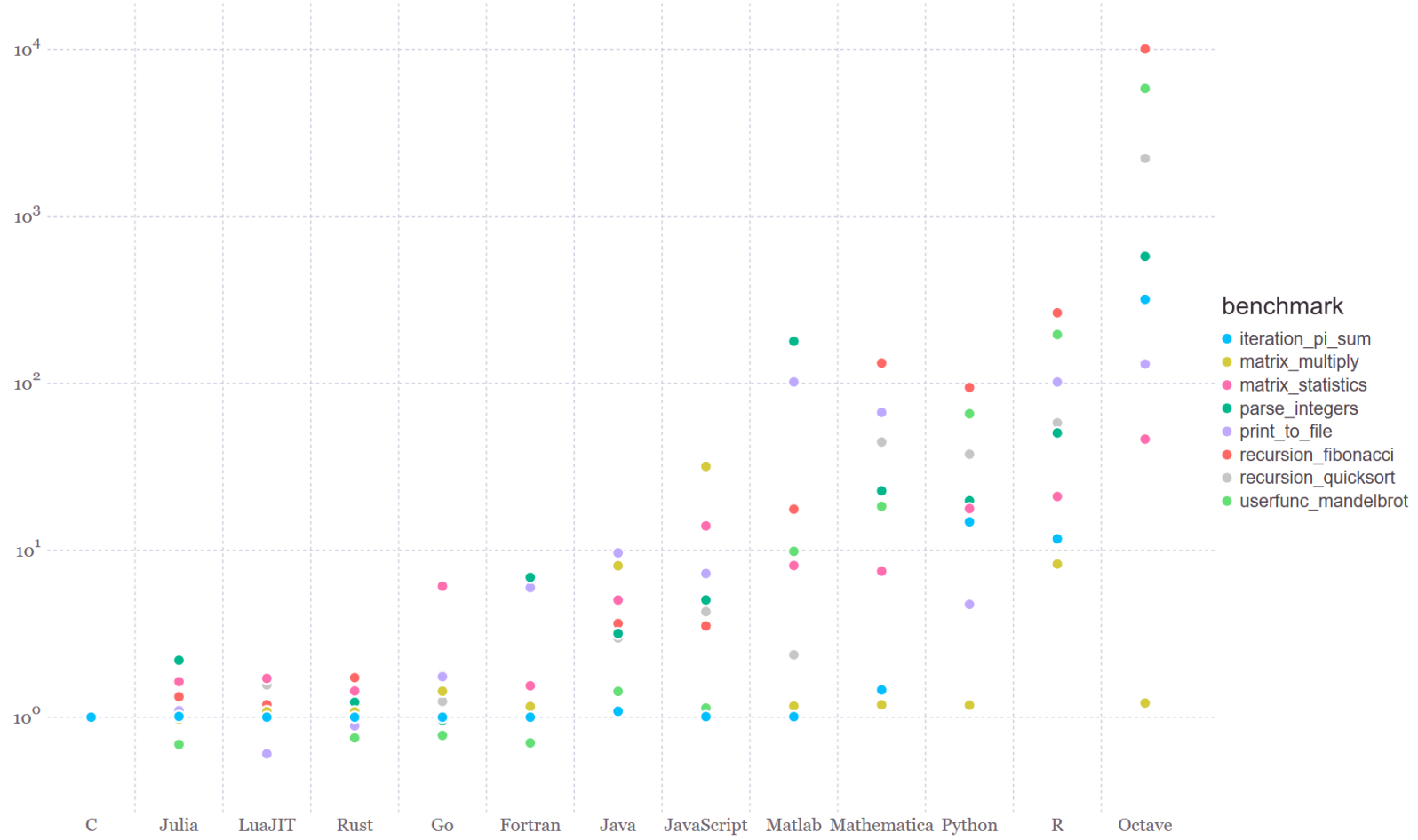


Developer

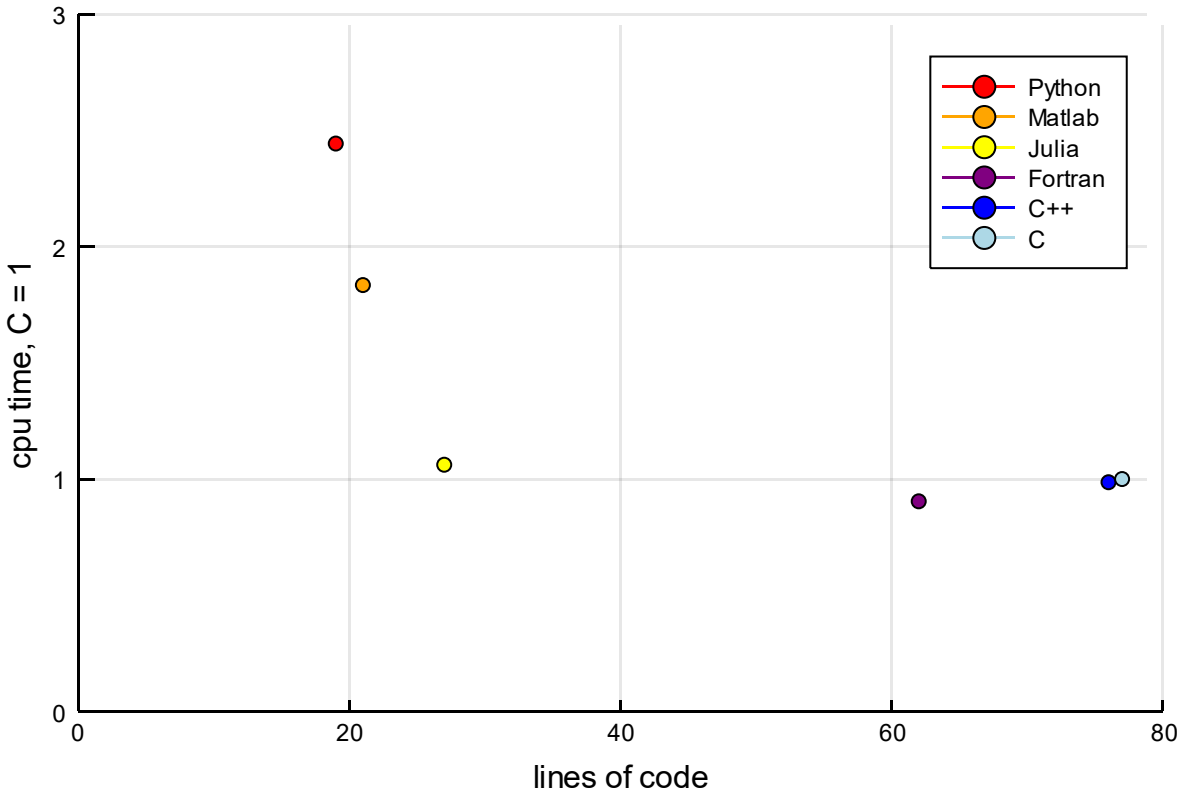
User



It is fast


10^4
10^3
10^2
10^1
10^0
C
Julia
LuaJIT
Rust
Go
Fortran
Java
JavaScript
Matlab
Mathematica
Python
R
Octave
benchmark
iteration_pi_sum
matrix_multiply
matrix_statistics
parse_integers
print_to_file
recursion_fibonacci
recursion_quicksort
userfunc_mandelbrot

It is expressive



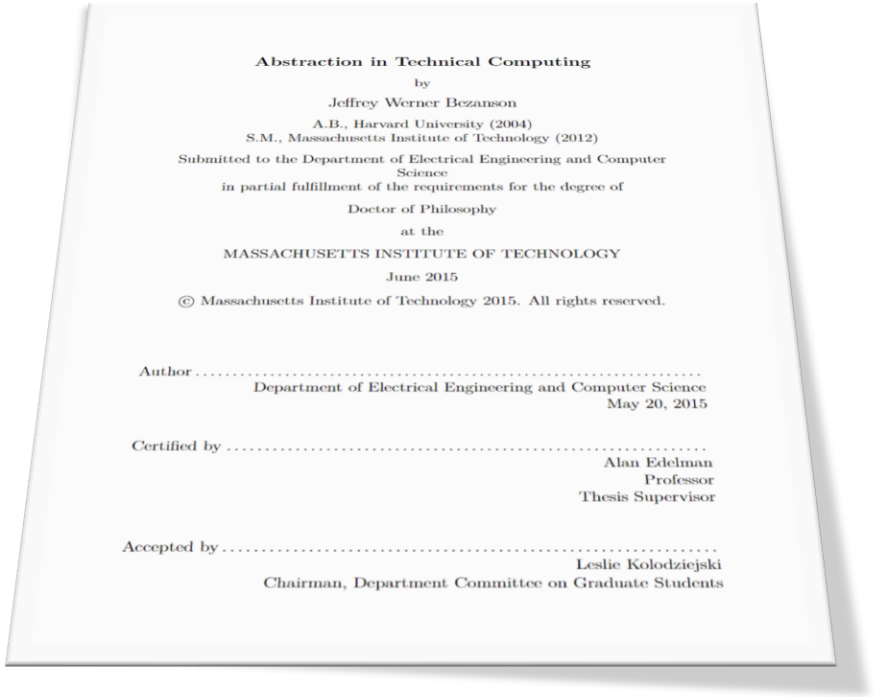
How old is Julia?

2009 Jeff, Stefan, Viral, and Alan start working on Julia

2012 Blog post: [Why we created Julia]

2015 Jeff's PhD & JuliaComputing

2018 Julia 1.0



Abstraction in Technical Computing

by

Jeffrey Werner Bezanson

A.B., Harvard University (2004)
S.M., Massachusetts Institute of Technology (2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Author .....
Department of Electrical Engineering and Computer Science
May 20, 2015

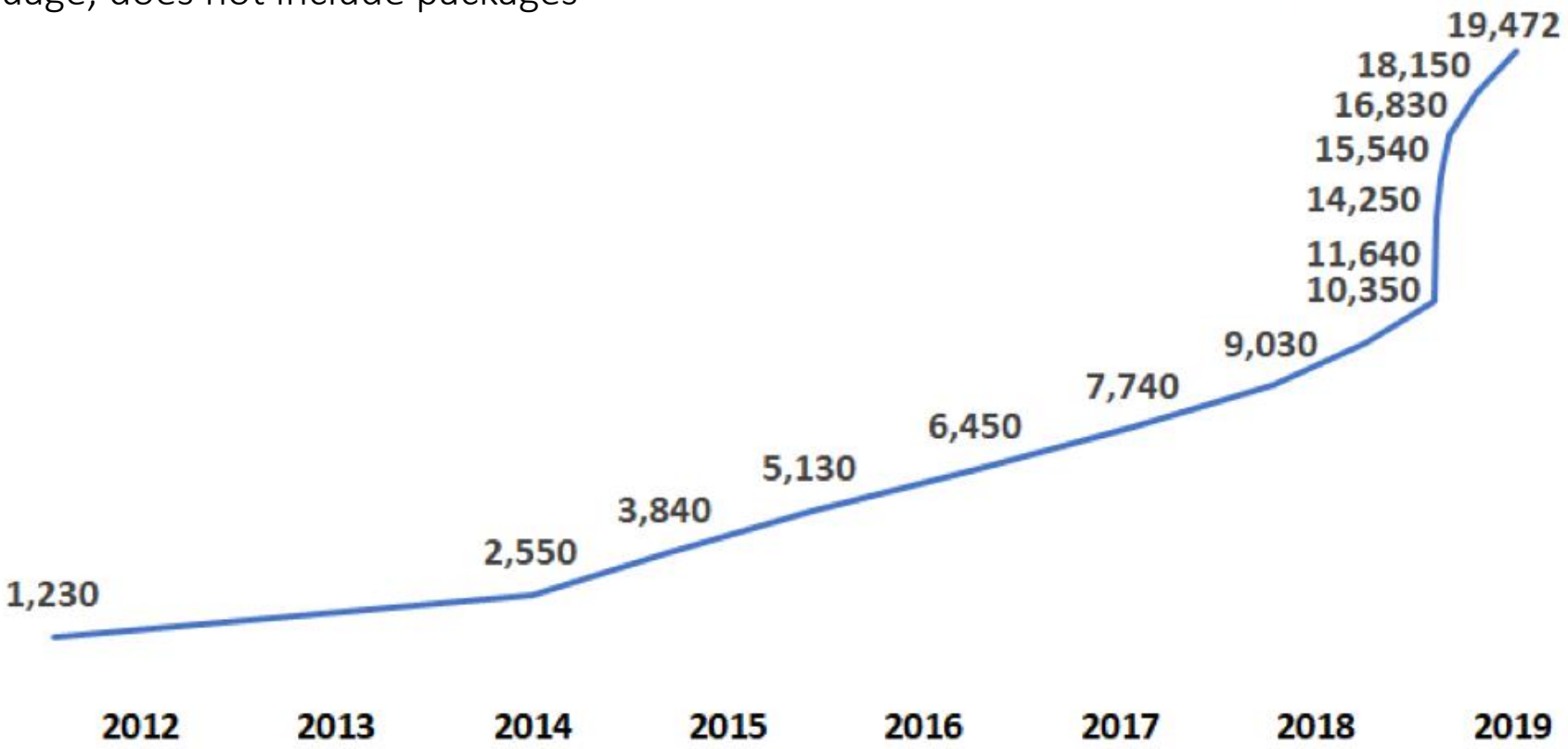
Certified by .....
Alan Edelman
Professor
Thesis Supervisor

Accepted by .....
Leslie Kolodziejski
Chairman, Department Committee on Graduate Students

Founders talk @ JuliaCon 2018: https://www.youtube.com/watch?v=1jN5wKvN-Uk

Julia GitHub stars

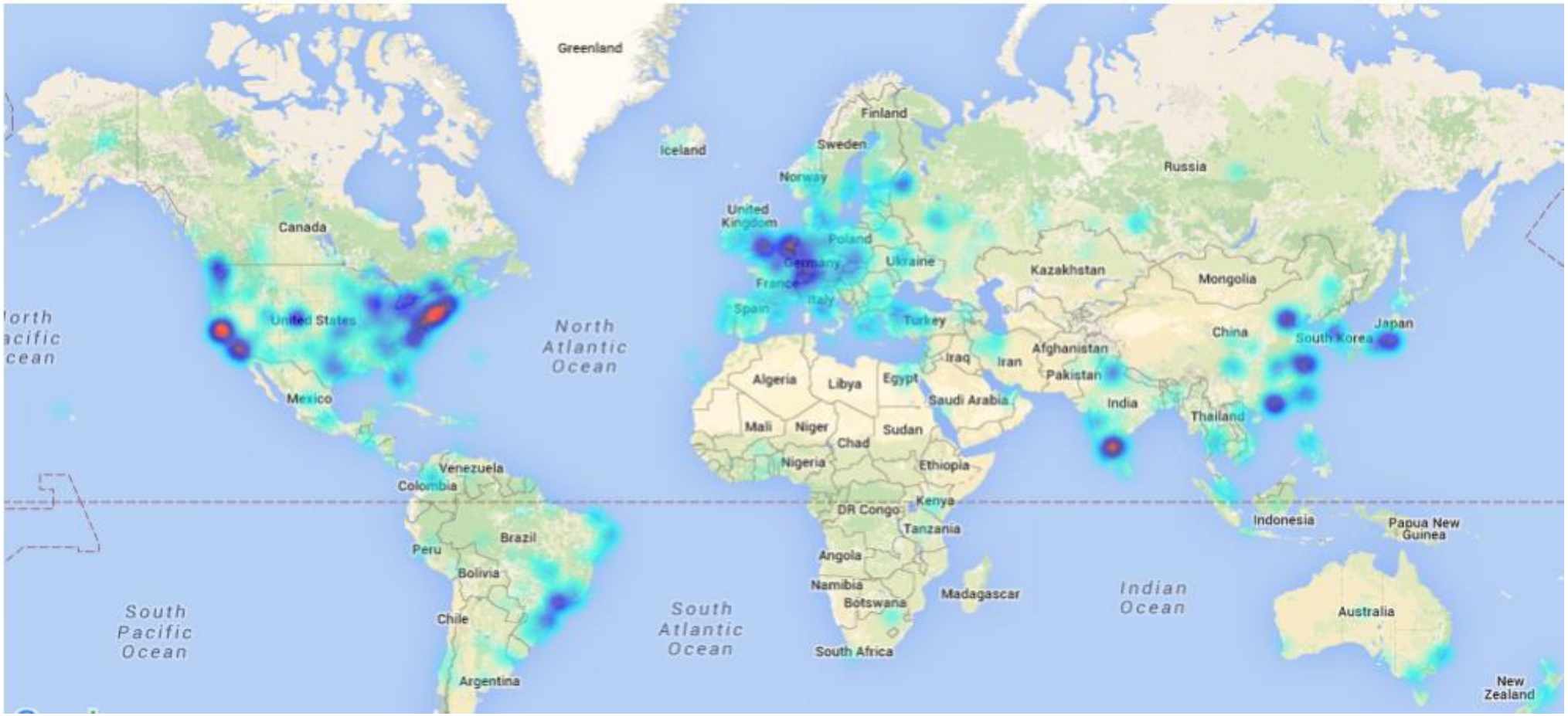
* Base language, does not include packages



https://juliacomputing.com/blog/2019/01/04/january-newsletter.html

A global community

More than 3 Million downloads, 2500 packages



James H. Wilkinson Prize For Numerical Software	Forbes 30 under 30	IEEE Babbage Prize IEEE Fellow
Stefan Karpinski Viral B. Shah Jeff Bezanson (2019)	Keno Fischer (2019)	Prof. Alan Edelman (2018)
		

BROWN UNIVERSITY

CORNELL UNIVERSITY FOUNDED A.D. 1865

EMORY UNIVERSITY

NORTHEASTERN UNIVERSITY
LVX VERITAS VIRTVS
FOUNDED A.D. 1898
BOSTON, MASSACHUSETTS

PSU
PENN STATE UNIVERSITY

PURDUE

SZKOŁA GŁÓWNA HANDLOWA W WARSZAWIE

Stanford University

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

THE UNIVERSITY OF EDINBURGH

UNIVERSITY of GLASGOW

UNIVERSITY of SOUTH FLORIDA 1956
TRUTH WISDOM

UNIVERSITE PAUL SABATIER
TOULOUSE III

UNIVERSITY OF CONNECTICUT 1881

FIDE SPLENDET ET SCIENTIA

THE UNIVERSITY OF CALIFORNIA BERKELEY
LET THERE BE LIGHT
1868

Western
UNIVERSITY · CANADA

ARIZONA STATE UNIVERSITY 1885
DITAT DEUS

MŰEGYETEM 1782

ILLINOIS STATE UNIVERSITY 1857
Gladly we learn and teach

SIDERE MENS EADEM MUTATO

CUNY THE CITY UNIVERSITY OF NEW YORK

EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MIT

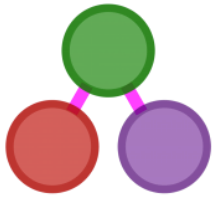
TOKYO METROPOLITAN UNIVERSITY
首都大学東京

Ucla

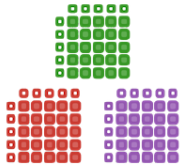
AGH

Best in class packages

Differential Equations



Graph Processing



Data Science

Image Processing

Deep Learning

Mathematical Optimization

Signal Processing

Computational Biology

Recommended talks

Nick Eubank: What Julia Offers Academic Researchers

George Datseris: Why Julia is the most suitable language for science