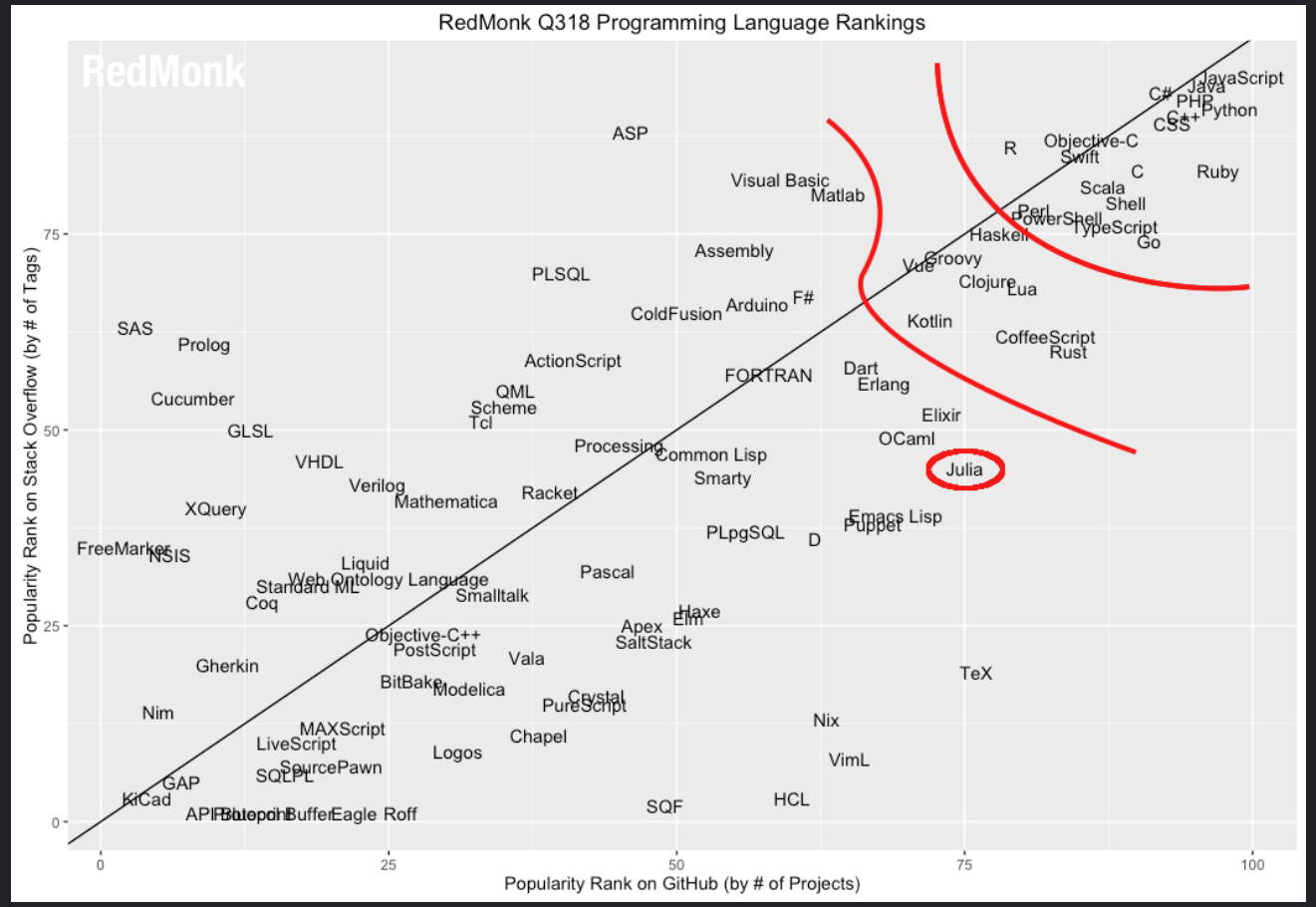

# So you're interested in Julia?

Cédric Simal

24/03/2020


Julia computing
PRODUCTS
INDUSTRIES
CASE STUDIES
TRAINING
COMMUNICATION
RESOURCES
BLOG
Here is the latest from Julia Computing
BLOG
Julia Computing Goes Wild with Career Zoo in Limerick, Ireland
14 Jan 2020 | Julia Computing
Limerick, Ireland – Julia Computing will participate in Career Zoo's 'Tech on the Wild Atlantic Way' event in Limerick, Ireland on February 1, 2020.
Julia Computing's Avik Sengupta (VP Engineering) will lead a workshop on 'Machine Learning with Julia'.
'Tech on the Wild Atlantic Way' is the premier recruiting, networking and training event in Ireland for tech, biotech, aviation and engineering firms and workers.
Julia is the fastest high-performance open source computing language for data, analytics, algorithmic trading, machine learning, artificial intelligence, and other scientific and numeric computing applications. Julia solves the two language problem by combining the ease of use of Python and R with the speed of C++.
RECENT POSTS
Newsletter March 2020
02 Mar 2020 | Julia Computing
Automatic Differentiation Meets Conventional Machine Learning
24 Feb 2020 | Deepak Suresh and Abhijith Chandraprabhu
Newsletter February 2020
07 Feb 2020 | Julia Computing

# Despite rapid growth, Julia is still in early adoption




https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/

# Disclaimer

I am **not** a Julia expert

This talk is mostly about my experience learning Julia

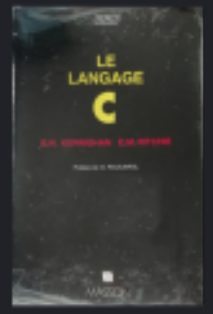
Links to proper learning resources will be provided

# What to expect

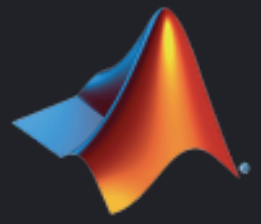
1. How I got into Julia
2. What is Julia, and why is it fast?
3. Julia's Ecosystem
4. Demo

# My programming history

2014 2015 2016 2017 2018 2019 2020

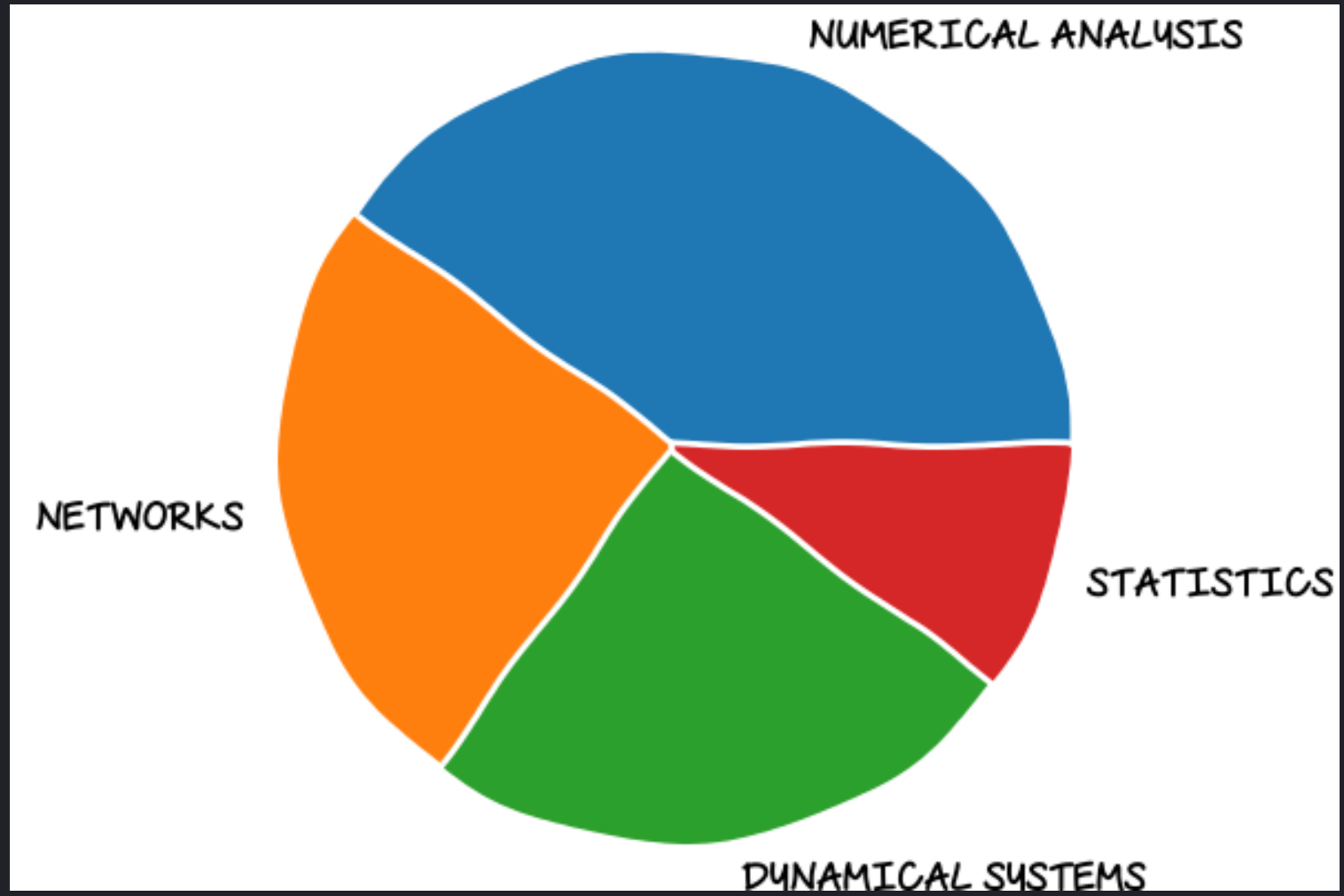




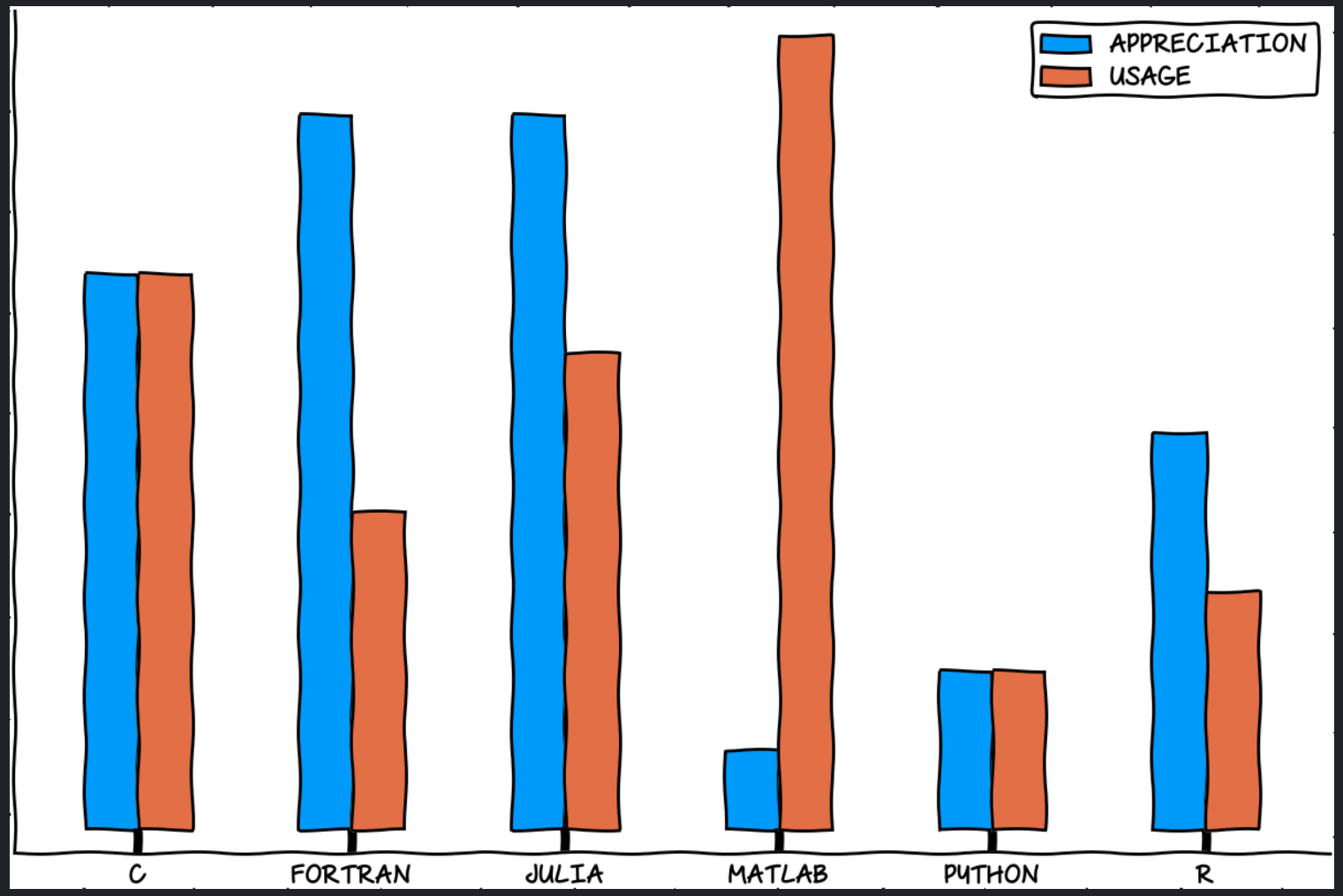





# Mathematical interests



# Languages



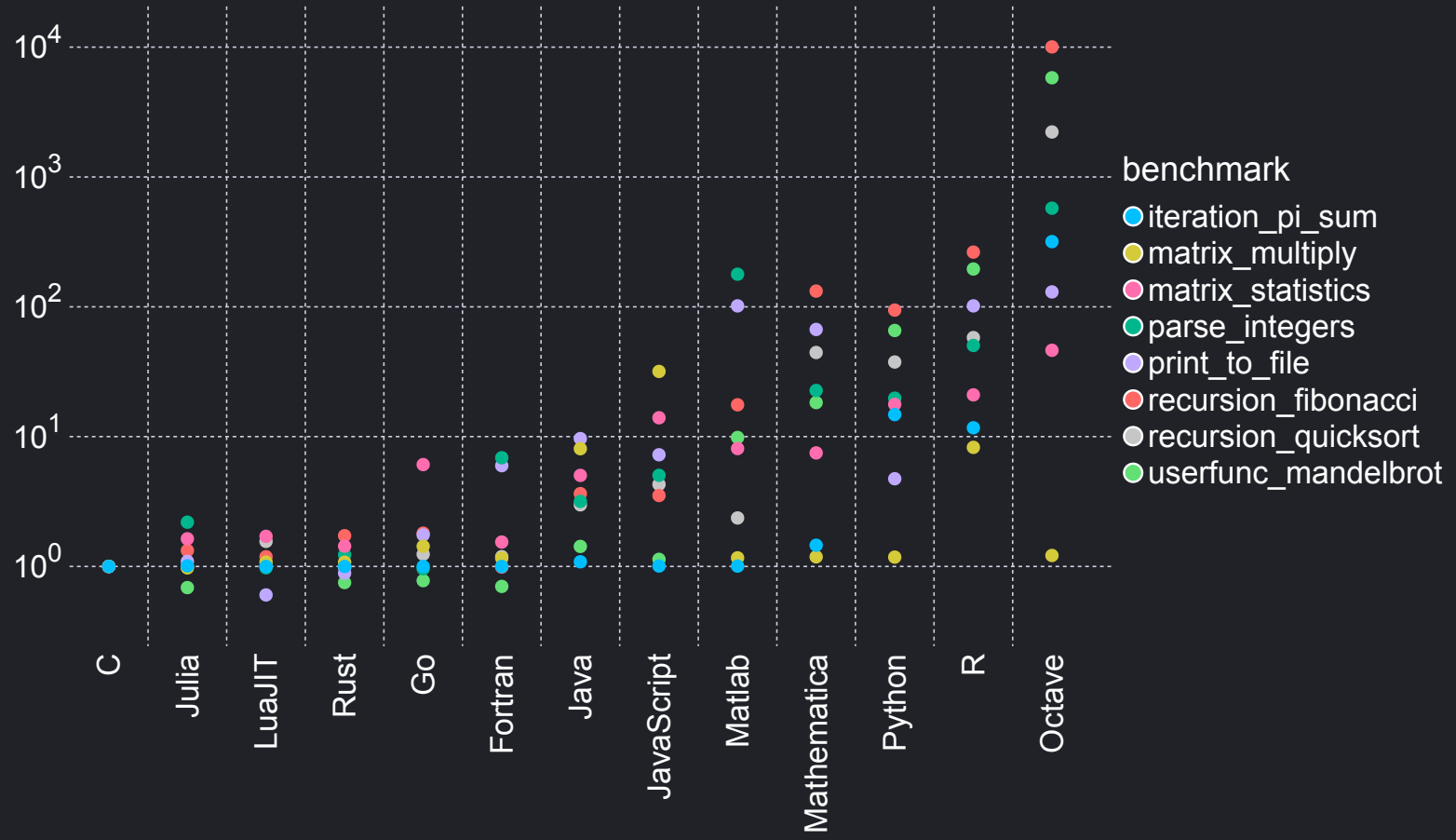
# My programming philosophy

- Performance matters
- Elegance matters
- The right tool for the right job

# The Two Languages Problem

| C/C++/Fortran | Python/R/Matlab |
| --- | --- |
| Compiled | Interpreted |
| Static Typing | Dynamic Typing |
| Limited Abstraction | Flexible |

# Performance




https://julialang.org/benchmarks/

# Parallel computing

- Difficult in traditional languages (C, Fortran, ...)
- Naturally easy in Functional Programming (Erlang, Haskell)
- New generation of "hybrid" languages (Go, Julia, Rust, ...)

# Meet Julia

## Julia: A Fast Dynamic Language for Technical Computing

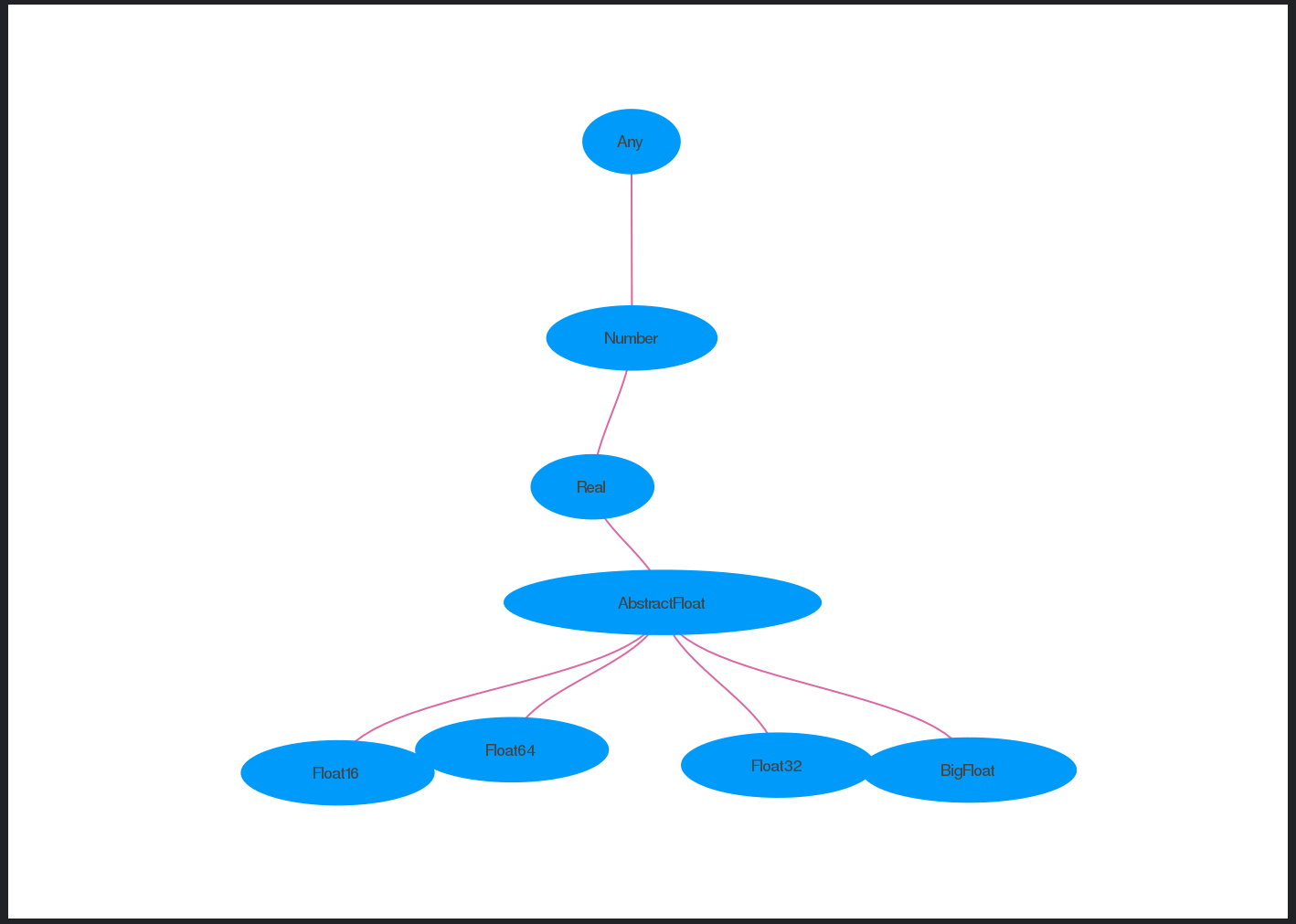
Jeff Bezanson* MIT
Stefan Karpinski† MIT
Viral B. Shah‡
Alan Edelman§ MIT

September 25, 2012

**Abstract**

Dynamic languages have become popular for scientific computing. They are generally considered highly productive, but lacking in performance. This paper presents Julia, a new dynamic language for technical computing, designed for performance from the beginning by adapting and extending modern programming language techniques. A design based on generic functions and a rich type system simultaneously enables an expressive programming model and successful type inference, leading to good performance for a wide range of programs. This makes it possible for much of Julia's library to be written in Julia itself, while also incorporating best-of-breed C and Fortran libraries.

# Types


Any
Number
Real
AbstractFloat
Float16
Float64
Float32
BigFloat

# Type System

Julia's type system is

- Dynamic, with optional type annotations `x::Int`
- Parametric `Vector{T}`
- Hierarchical (subtyping) `Float64 <: Real`

# Multiple Dispatch

```
julia> +
+ (generic function with 198 methods)

julia> methods(+)
# 198 methods for generic function "+":
[1] +(x::Bool, z::Complex{Bool}) in Base at complex.jl:282
[2] +(x::Bool, y::Bool) in Base at bool.jl:96
[3] +(x::Bool) in Base at bool.jl:93
[4] +(x::Bool, y::T) where T<:AbstractFloat in Base at bool.jl:104
[5] +(x::Bool, z::Complex) in Base at complex.jl:289
[6] +(a::Float16, b::Float16) in Base at float.jl:398
[7] +(x::Float32, y::Float32) in Base at float.jl:400
[8] +(x::Float64, y::Float64) in Base at float.jl:401
[9] +(z::Complex{Bool}, x::Bool) in Base at complex.jl:283
[10] +(z::Complex{Bool}, x::Real) in Base at complex.jl:297
...
```

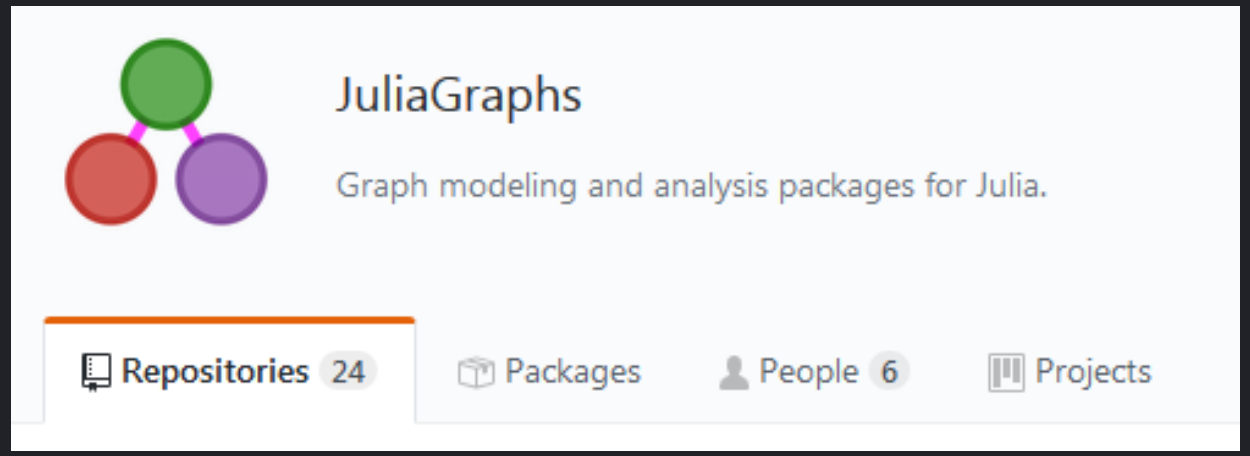
# The secret sauce behind Julia's speed

Type system + Multiple dispatch + JIT compiling

# Postcard Demo!

# Packages

- Everything on Github (easy to create and publish)
- Standardized structure and testing
- Open source




# Networks

LightGraphs.jl

# Numerical Integration

JuliaDiffEq

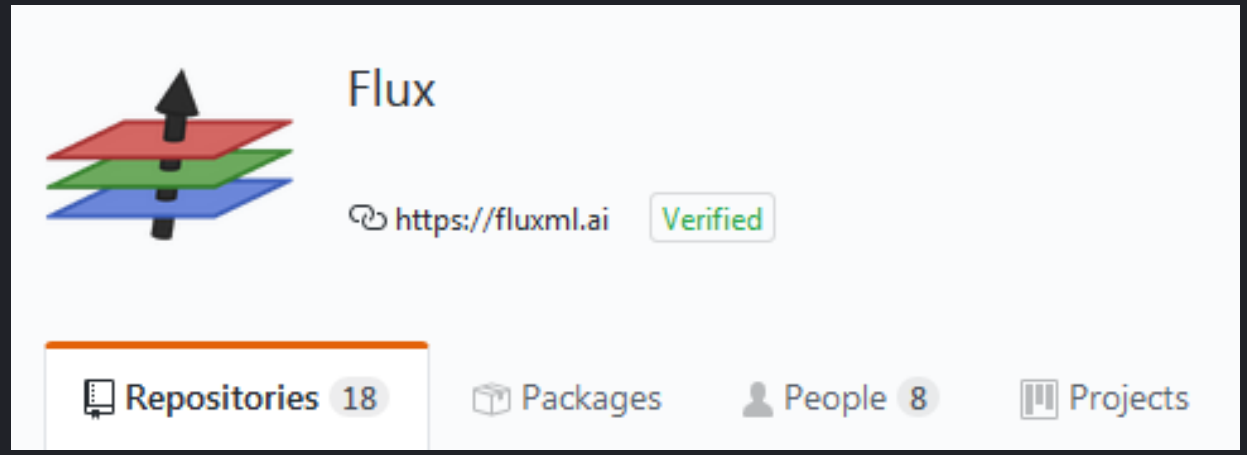

Friendship ended with
MATLAB
Now
julia
is my
best friend
imgflip.com

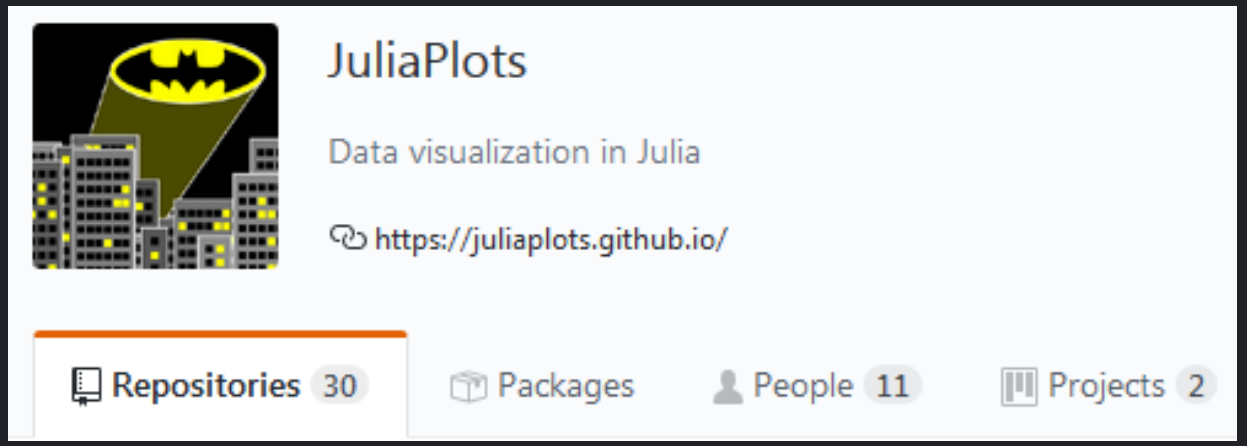

# Statistics

JuliaStats

# Machine Learning

- Flux
- TensorFlow.jl



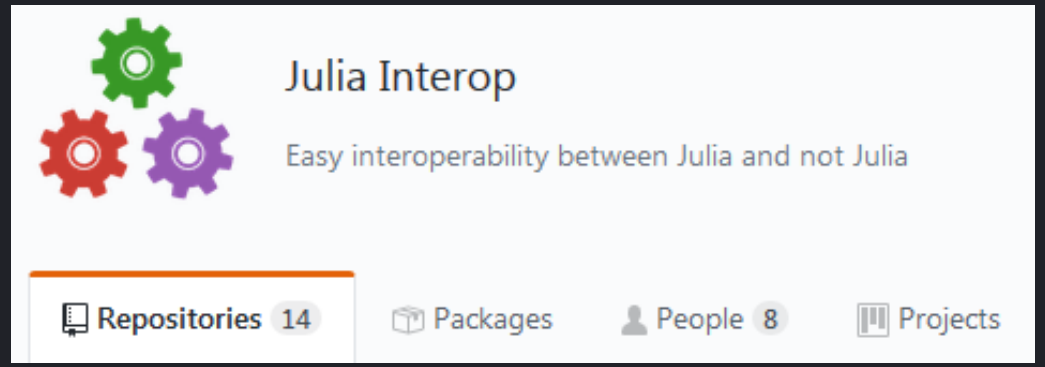



# Plotting

- Plots.jl
- Gadfly.jl

# Don't throw away your old code!

- Natively call C/Fortran functions
- PyCall.jl
- RCall.jl



# Demo Time!

# What's great about Julia

- Performance
- Composability
- Good practices
- Open source

# What's not so great about Julia

- Not quite mature
- Slow startup ("Time to first plot")
- Performance > Abstraction/Correctness

# Why Julia might not be for you

- Still in development
- *Programs*, not scripts

# That's all folks!

Slides, code and *many* links at
https://github.com/csimal/JuliaTalk

Send questions/comments/complaints to
cedric.simal@student.unamur.be