

So you heard about Julia?

Cédric Simal

Unamur, Naxys

15/12/21

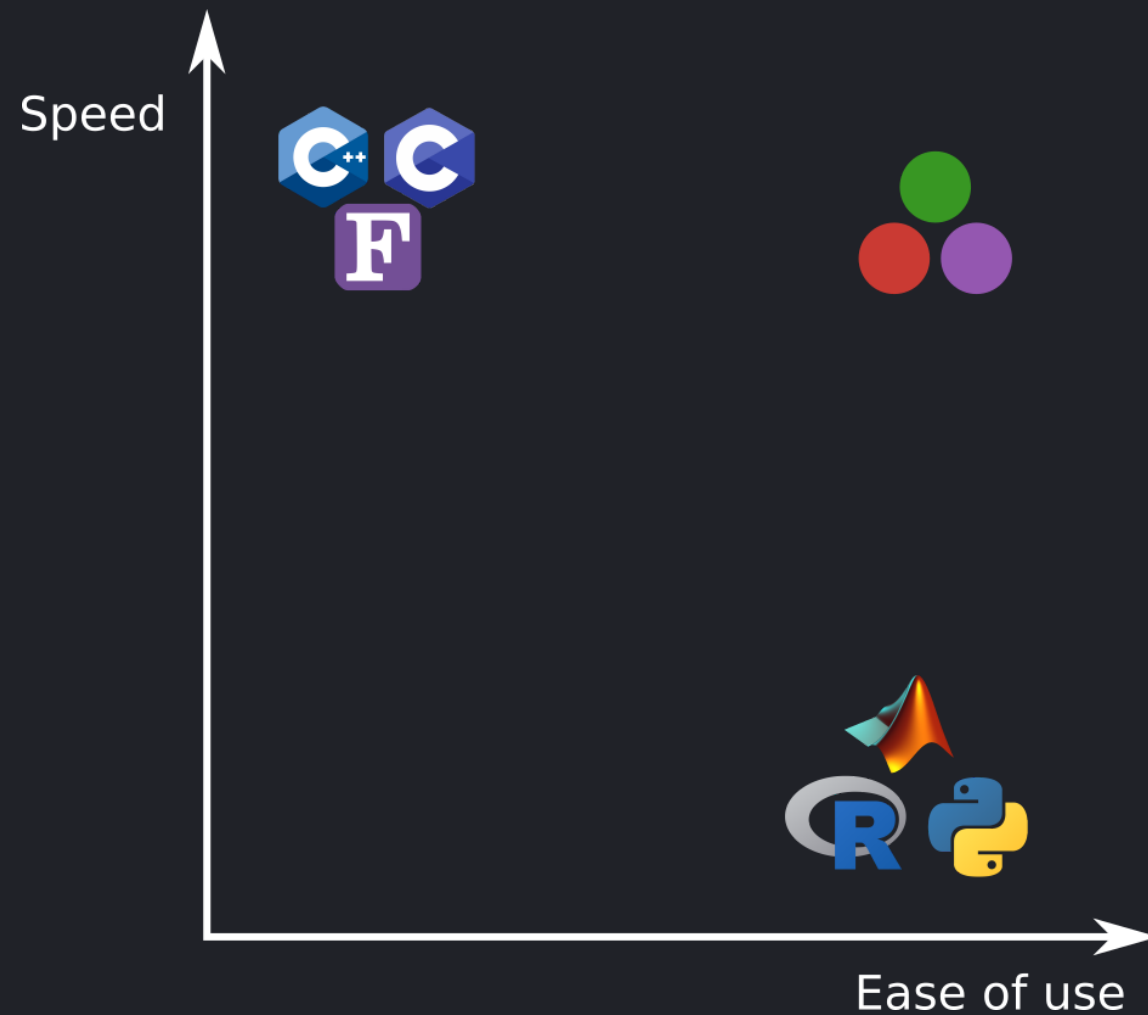


Motivation

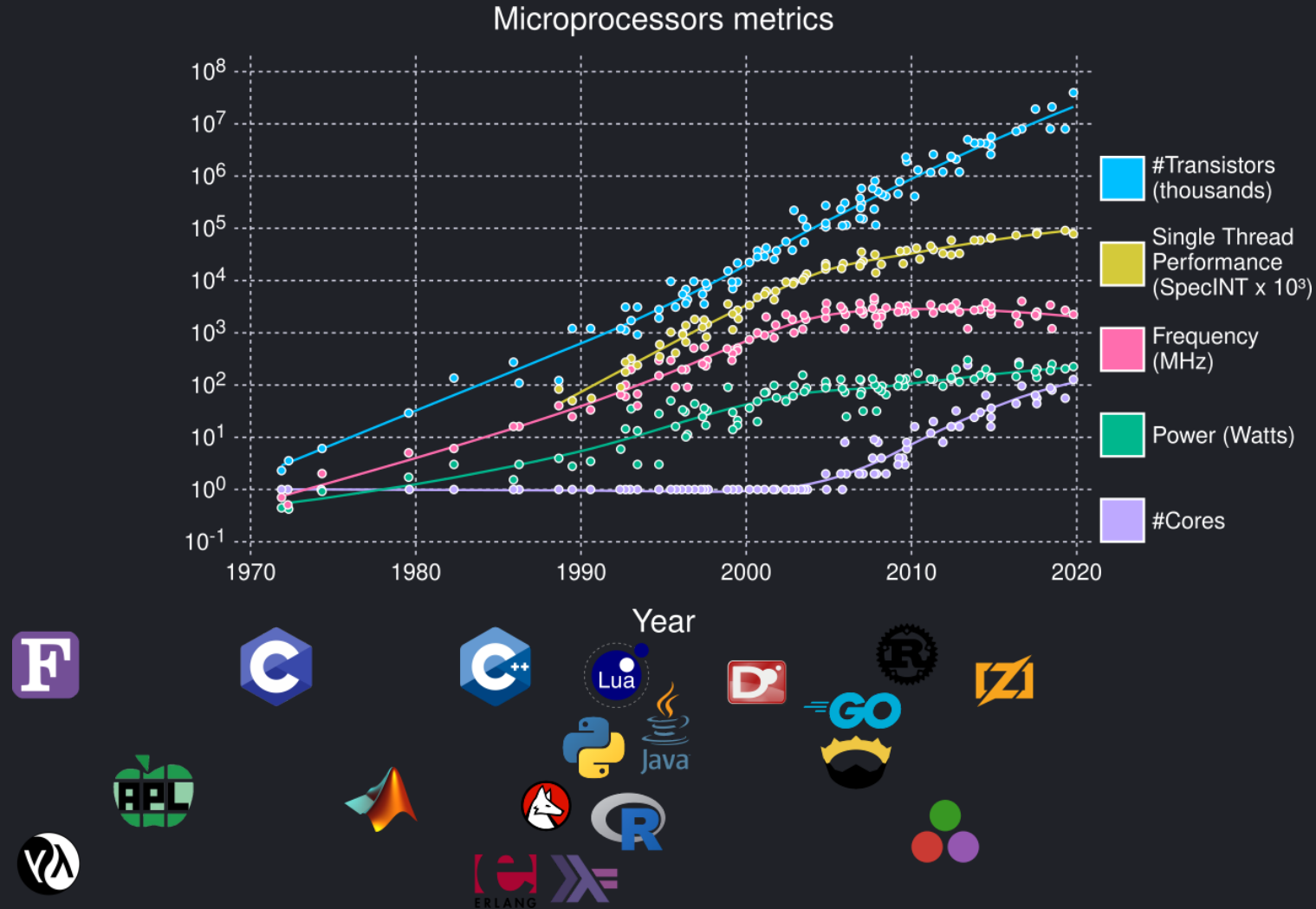
Outline

1. Modern problems in scientific computing
2. What is Julia and how does it work
3. The Julia package ecosystem

The two languages problem



The rise of parallel computing



data: <https://github.com/karlrupp/microprocessor-trend-data>

Meet Julia

Julia: A Fast Dynamic Language for Technical Computing

Jeff Bezanson*
MIT

Stefan Karpinski†
MIT

Viral B. Shah‡

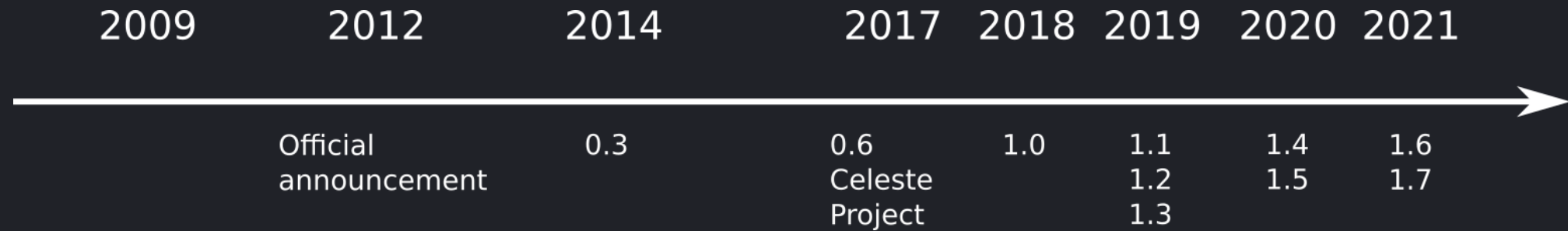
Alan Edelman§
MIT

September 25, 2012

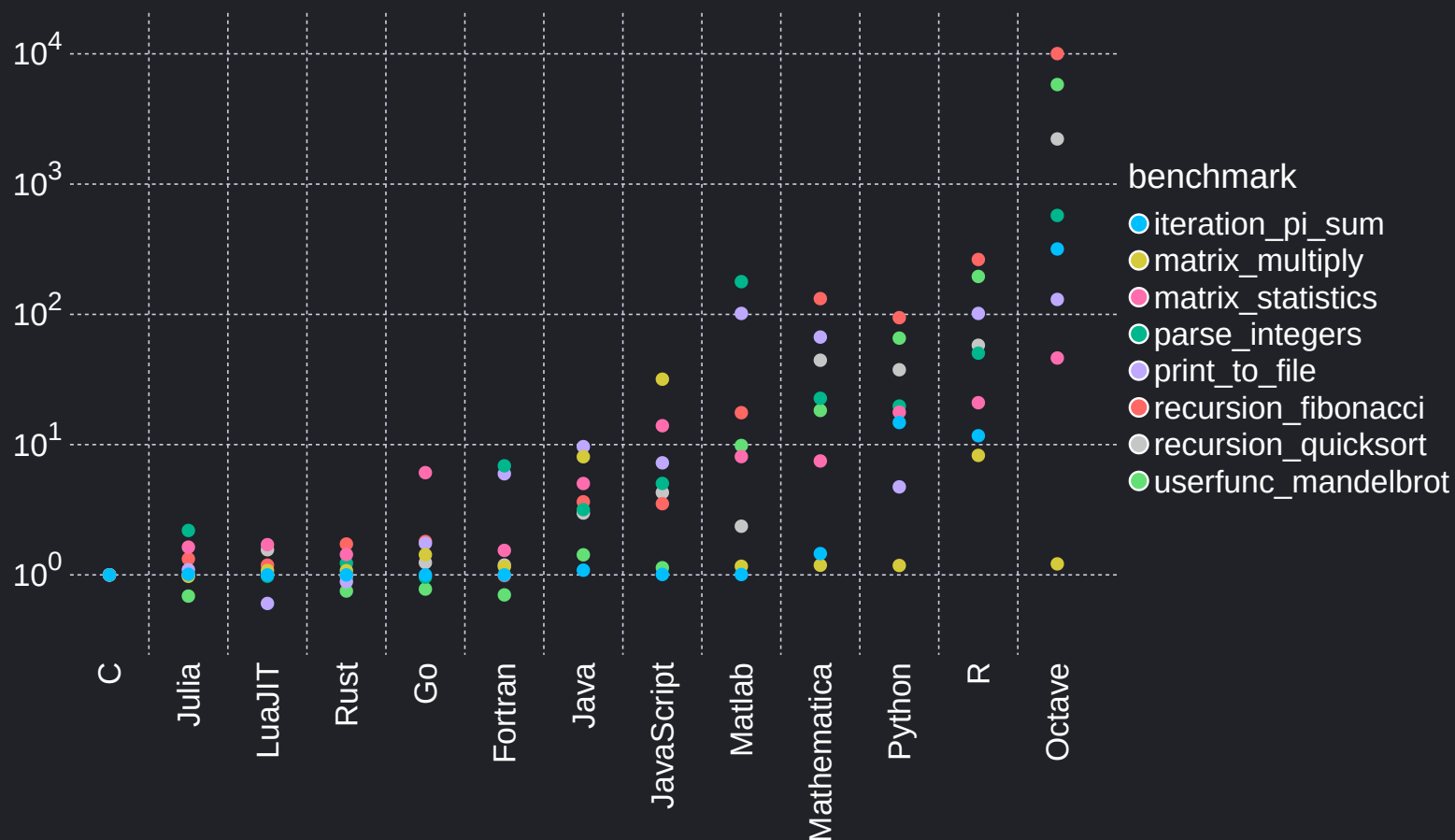
Abstract

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.

A short history of Julia

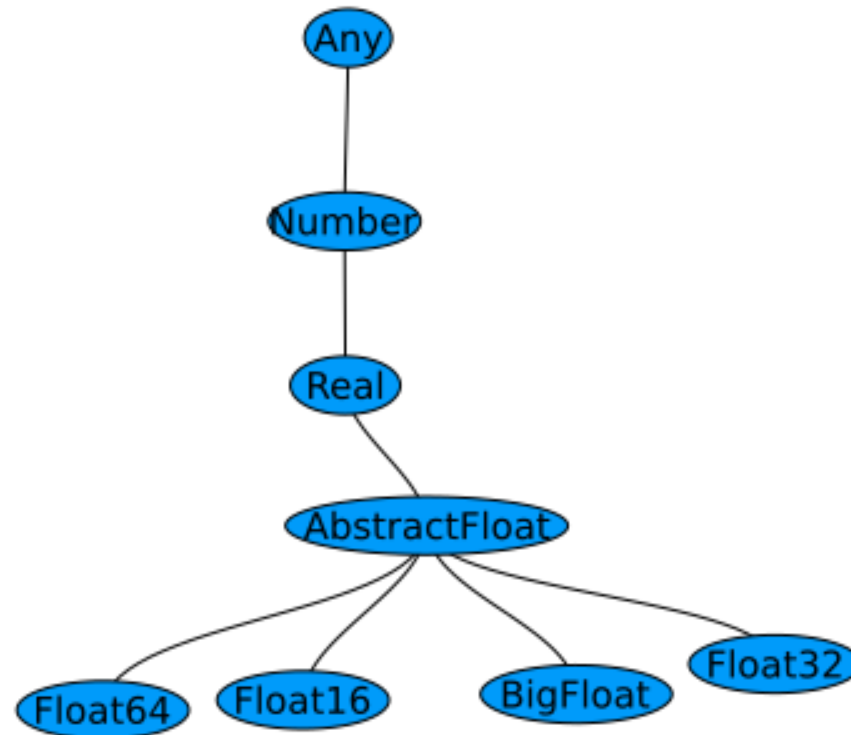


Julia is fast



<https://julialang.org/benchmarks/>

Types



Type System

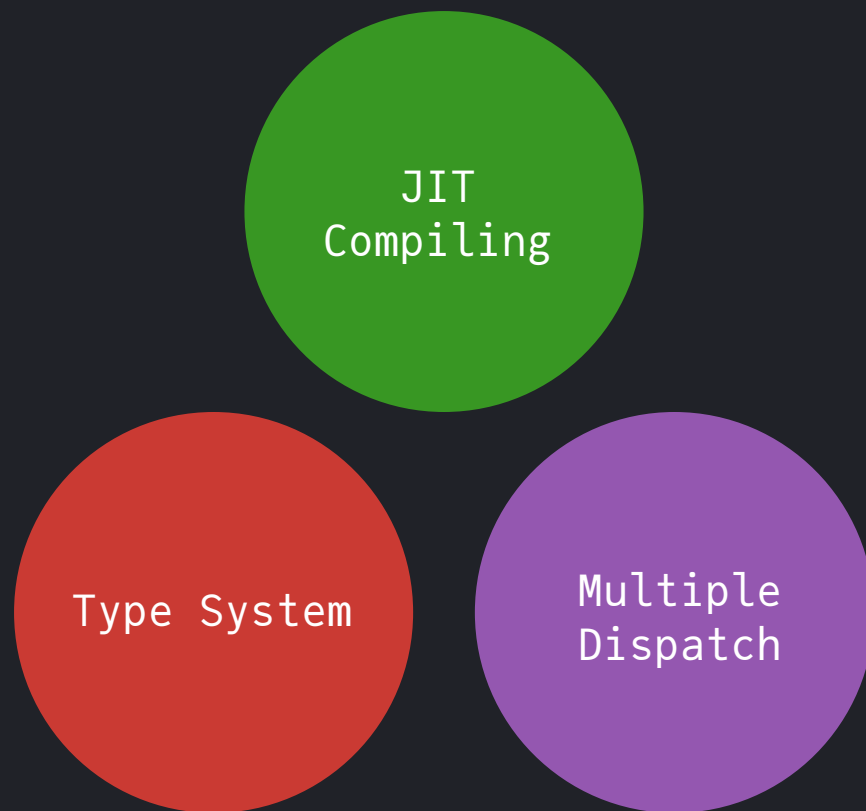
Julia's type system is

- 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



Postcard demo

Dynamical Systems

- DifferentialEquations
- DynamicalSystems



SciML Open Source Scientific Machine Learning

Open source software for scientific machine learning

<https://sciml.ai> [@SciML_Org](https://twitter.com/SciML_Org) contact@chrisrackauckas.com

[Overview](#)

[Repositories](#) **117**

[Packages](#)

[People](#) **35**

[Projects](#)



Networks





JuliaGraphs


Graph modeling and analysis packages for the Julia programming language


<https://juliagraphs.org>


Verified

 Overview

 Repositories 31

 Packages

 People 6

 Projects

- Graphs
- NetworkDynamics
- SimpleHypergraphs
- Simplicial

Statistics





Julia Statistics


Statistics and Machine Learning made easy in Julia


<https://juliastats.org>


Verified

 Overview

 Repositories 40

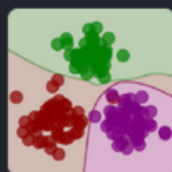
 Packages

 People 18

 Projects



Machine Learning



JuliaML

Julia packages for Machine Learning

<https://juliaml.github.io>

[Overview](#) [Repositories 28](#) [Packages](#) [People 13](#) [Projects](#)



JuliaAI

Home for repositories of the MLJ (Machine Learning in Julia) project

<https://github.com/alan-turing-institut...>

[Overview](#) [Repositories 35](#) [Packages](#) [People 3](#) [Projects](#)

Physics

QuantumOptics


Yao





JuliaPhysics


Physics in Julia


<http://juliaphysics.github.io>

 **Overview**


 Repositories **12**

 Packages

 People **6**

 Projects

Astronomy & Aerospace



Julia Astro

<http://JuliaAstro.github.io> julia-astro@googlegroups.com

[Overview](#) [Repositories 38](#) [Packages](#) [People 11](#) [Projects](#)

Optimization

Optim




GalacticOptim





Julia Interop

Easy interoperability between Julia and not Julia

 **Overview**  Repositories **16**  Packages  People **12**  Projects

Interop

Plotting


- Plots
- Gadfly
- Makie





JuliaPlots


Data visualization in Julia


<https://juliaplots.github.io/>

 Overview

 Repositories 38

 Packages

 People 20

 Projects 2

Conclusions

- Compared to a year ago, Julia has grown tremendously
- It delivers on its promise to "Walk like Python. Run like C"
- Main challenge: reach critical mass

That's all folks!

The Virgin



Cringe puns
for package
names

0-based? More like
unbased!

Object Oriented

GIL

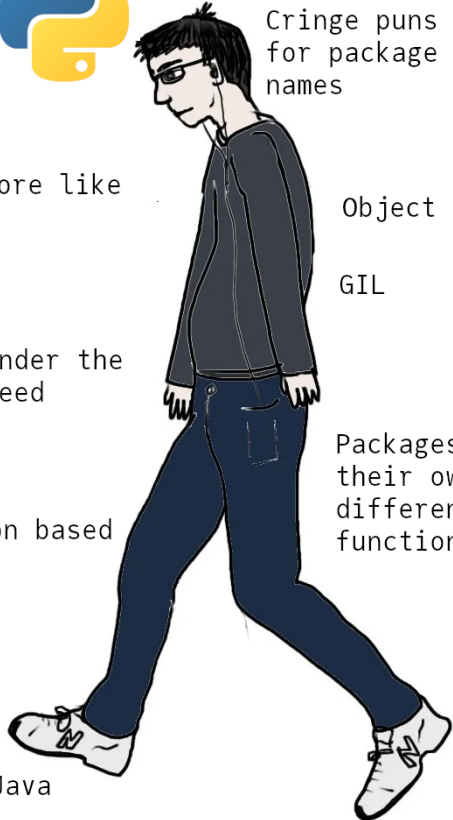
Needs C++ under the
hood for speed

Indentation based
scoping

Packages provide
their own
differentiable
functions

Older than Java

Python 2



THE CHAD



1-based and Fortran-
pilled

Multiple dispatch

Runs natively on CUDA
gpus

Pollutes main
namespace like a boss

Almost native
differentiable
programming

Metaprogramming

Code looks like maths

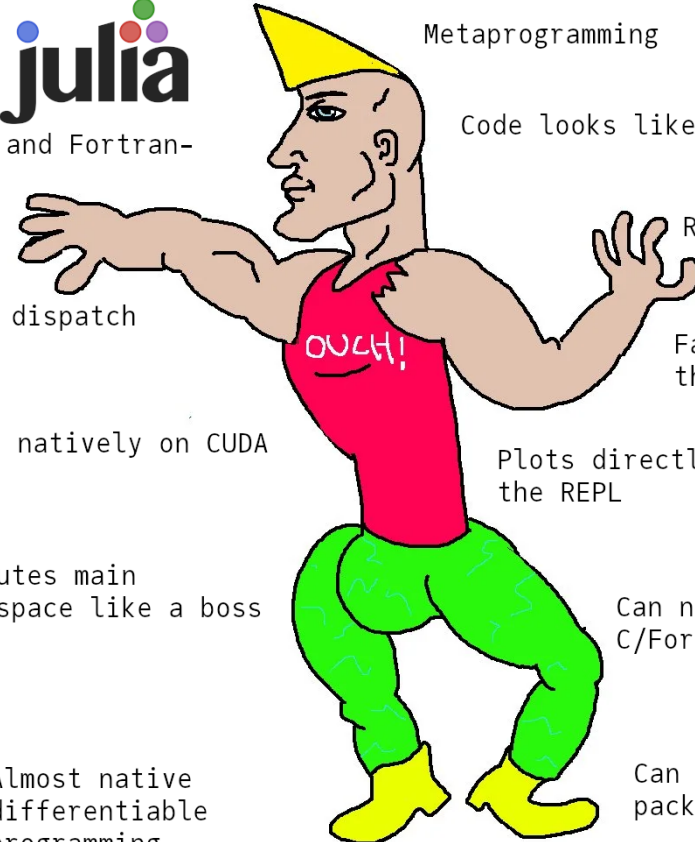
Reactive notebooks

Fastest ODE suite on
the market

Plots directly in
the REPL

Can natively call
C/Fortran code

Can load any Python
package using PyCall.jl



<https://github.com/csimal/Julia-Unamur>