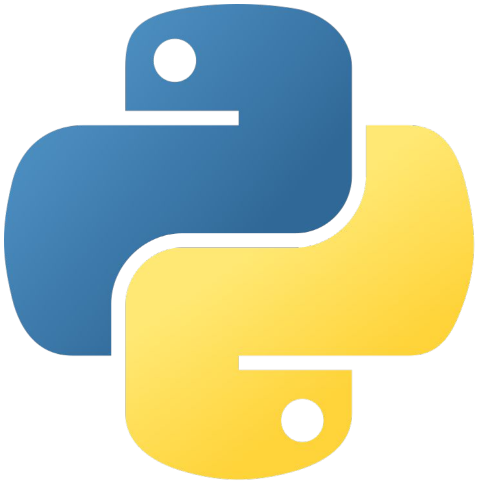
Perspective: The State of Julia for Scientific Machine Learning

Edward Berman, Jacob Ginesin
Northeastern University

Everyone uses Python!

- Python is…
  - ○ Convenient
  - ○ Universal
  - ○ Hackable
  - ○ Well-supported



We all love Python… right?

WRONG

Python Sucks.

Many reasons to hate Python!

- Slow!
- Hard to maintain!
- Pip is horrible!

Can we do better??

We want a language that's…

- FAST!
- Yet hackable

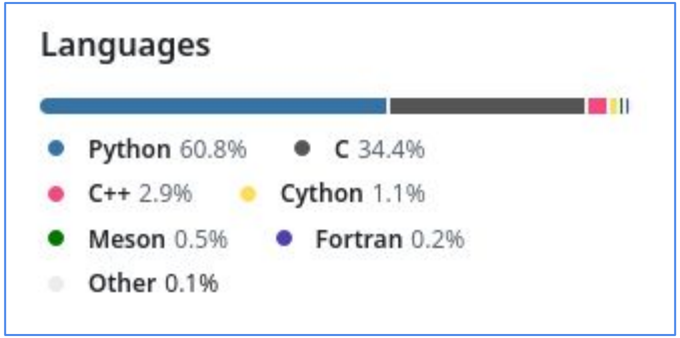
Can we do better??

This boils down to the…

“TWO LANGUAGE PROBLEM”

The Two Language Problem

- A language cannot be fast and high-level
- Python wraps C/C++ when performance is necessary
- (e.g. Torch, NumPy)



Languages

● Python 60.8%
● C 34.4%
● C++ 2.9%
● Cython 1.1%
● Meson 0.5%
● Fortran 0.2%
● Other 0.1%

[NumPy repo, Dec 2024]

Are there alternatives to Python!?

Our (Potential) Savior: julia !!!

- Claims to solve the two-language problem
- Fast **and** hackable

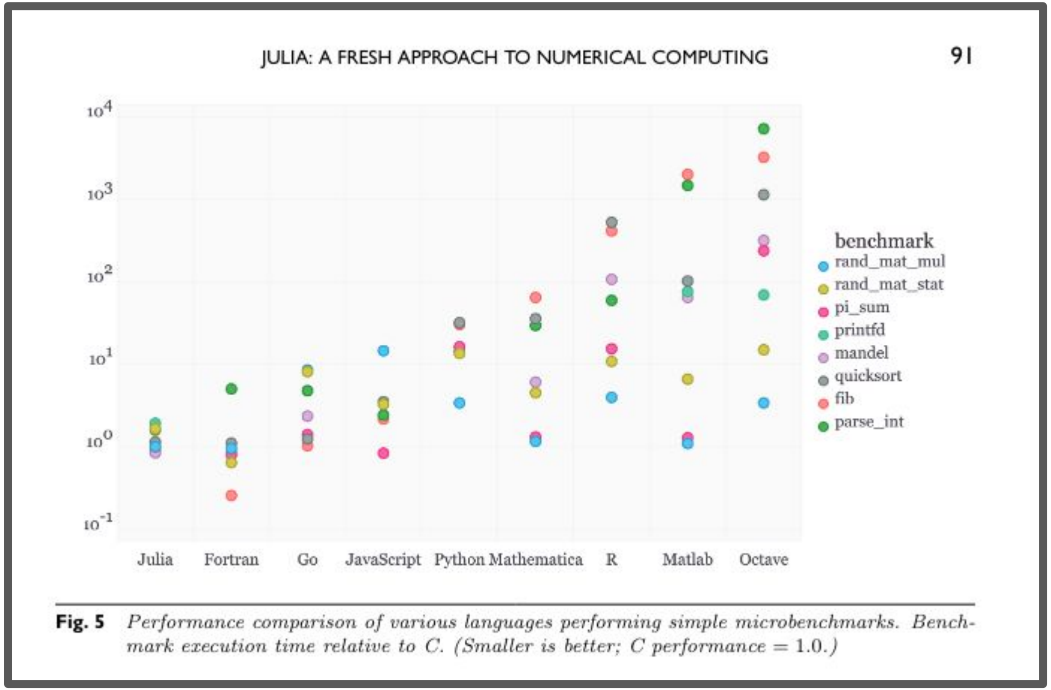
Our (Potential) Savior: julia !!!

- Julia defies the “laws of nature”:
  - High-level programs must be slow
  - Users must prototype in one language and deploy in another
  - Some things should be left for the experts

[Bezanson et al.]

Julia is Fast

- JIT Compiled
- Parallelism



JULIA: A FRESH APPROACH TO NUMERICAL COMPUTING 91

Fig. 5 *Performance comparison of various languages performing simple microbenchmarks. Benchmark execution time relative to C. (Smaller is better; C performance = 1.0.)*

(Bezanson et al.)

Julia is High-level

- Leverages multiple dispatch

Julia is High-level

- Leverages multiple dispatch
- Functional Paradigm

Julia is High-level

- Leverages multiple dispatch
- Functional Paradigm
- Garbage Collection

Julia is High-level

- Leverages multiple dispatch
- Functional Paradigm
- Garbage Collection
- Optional Type Annotations

Julia is High-level

- Leverages multiple dispatch
- Functional Paradigm
- Garbage Collection
- Optional Type Annotations
- Clear, thorough style guides [e.g. SciML]
    - “Stylish guides for stylish developers” :)

Julia’s great Ecosystem

- ML - SciML
- NLP - JuMP
- AD - JuliaDiff
- PPL - Turing
- DiffEq - DifferentialEquations
- PSF - ShOpt.jl
- More :)

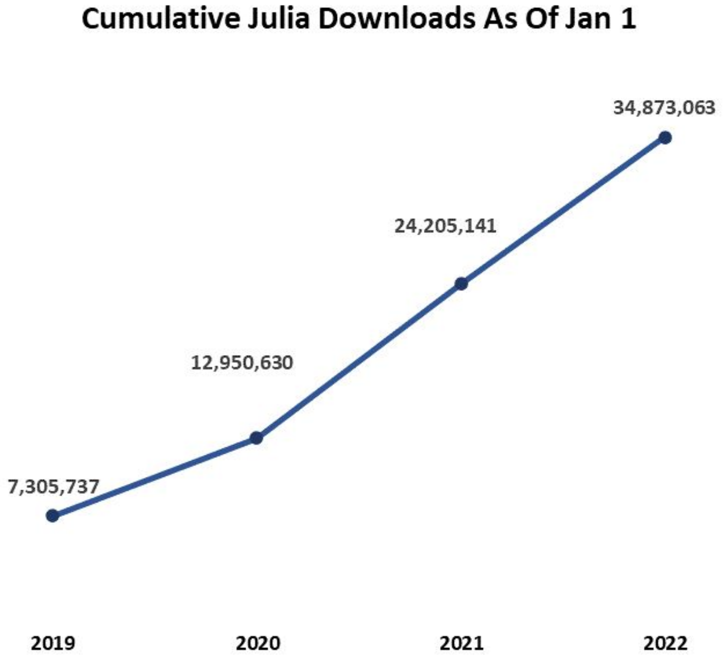
Julia’s great package manager (Pkg)

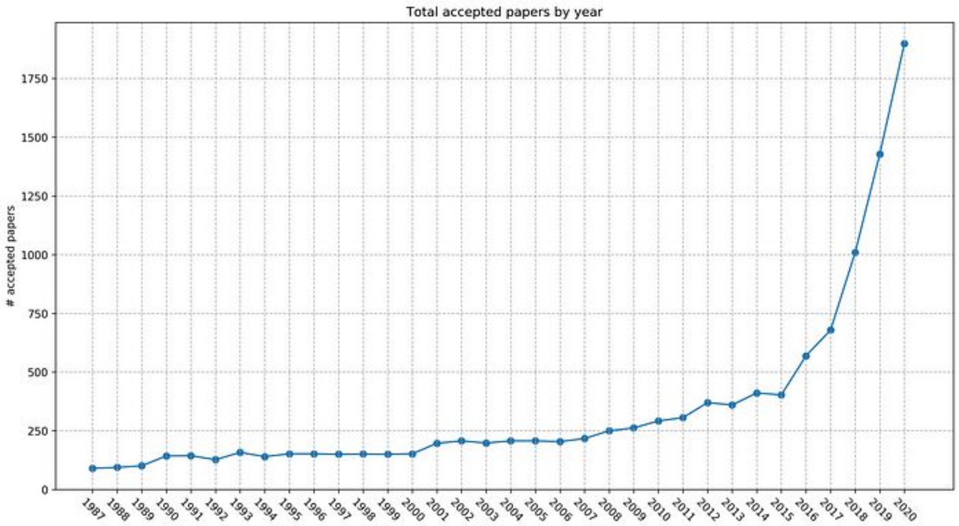
- REPL and environment integration :)

Growth

- Julia is great!
- It has a niche community
    - And… Julia is… growing?…

Julia Downloads and Accepted NeurIPS papers



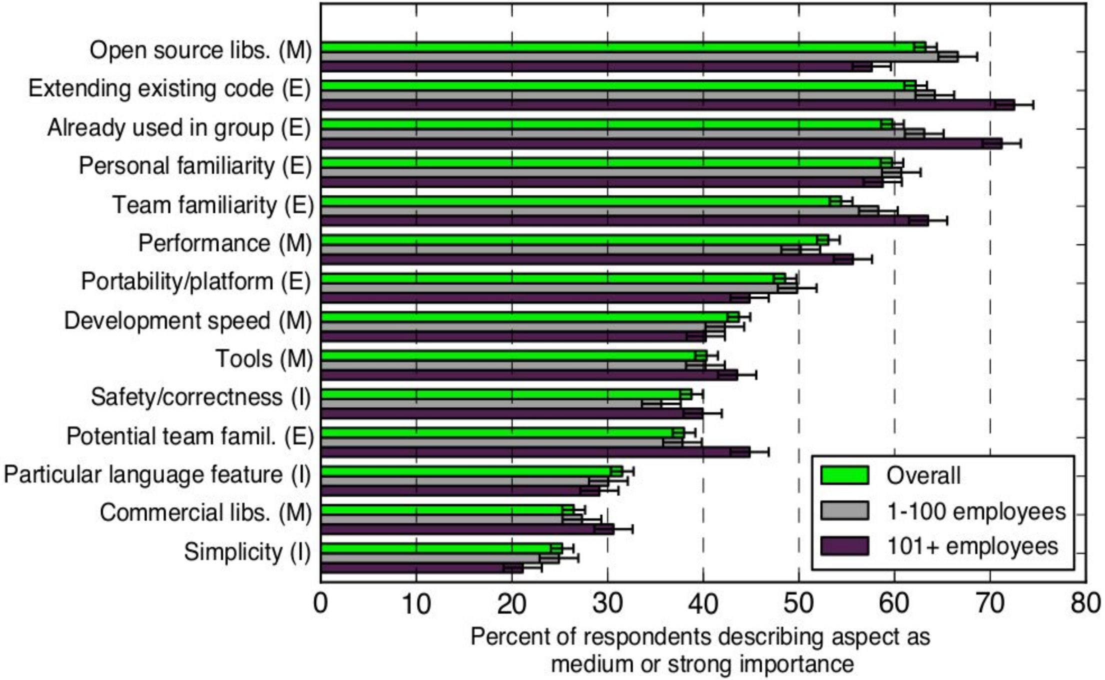


Momentum

- Why is Python still the de-facto language for Scientific Machine Learning Problems?
- Is the answer really just **momentum?**

Momentum

- Momentum plays some role in language adoption…



[meyerovich et al.]

Language Adoption

- So what gives?

Language Adoption

- So what gives?
- Julia has an amazing ecosystem with stellar performance

Language Adoption

- So what gives?
- Julia has an amazing ecosystem with stellar performance
- Julia has exquisite support for the sciences in particular

Language Adoption

- So what gives?
- Julia has an amazing ecosystem with stellar performance
- Julia has exquisite support for the sciences in particular
- PyTorch even has a Julia dependency! (PySR -> SymbolicRegressions.jl)

Language Adoption

- So what gives?
- Julia has an amazing ecosystem with stellar performance
- Julia has exquisite support for the sciences in particular
- PyTorch even has a Julia dependency! (PySR -> SymbolicRegressions.jl)
- **Why aren’t people using Julia?**

Julia Industry Support

- Industry Support (it’s the economy stupid!)
    - Julia doesn’t have nothing (case studies)

Julia Industry Support

- Industry Support (it’s the economy stupid!)
    - Julia doesn’t have nothing (case studies)
    - But it doesn’t compare to the competition

Julia Industry Support

- Industry Support (it’s the economy stupid!)
  - Julia doesn’t have nothing (case studies)
  - But it doesn’t compare to the competition

Google DeepMind

Julia Industry Support

- Users find this to be an issue
- Julia Hub Survey Data [JuliaCon 2024] finds:
    - 64% respondents said there are not enough Julia users in field or industry

Julia Industry Support

- Users find this to be an issue
- Julia Hub Survey Data [JuliaCon 2024] finds:
  - 64% respondents said there are not enough Julia users in field or industry
  - 71% use Julia for research, but only 16% for business critical tasks

Julia’s SWE infrastructure

- Julia lacks Testing and Engineering Features
  - Property-based testing, symbolic execution, contract-based testing, etc
- People want their software to be reliable!

Julia’s Interop

- Interoperability is also a big issue

Julia’s Interop

- Interoperability is also a big issue
- Most users will be coming from Python or C/C++

Julia’s Interop

- Interoperability is also a big issue
- Most users will be coming from Python or C/C++
- Backwards compatibility is paramount

Julia’s lack of maturity

- The 2024 JuliaCon survey also identified the **debugger, long stack traces, precompile times, and large executables** as problematic

Call to Action

- Julia has the potential to save us!

Call to Action

- Julia has the potential to save us!
- … but has withstanding issues

Call to Action

- Julia has the potential to save us!
- … but has withstanding issues
- **Think critically about the tools we use**

Try julia !!!

Fin!

- We explored Julia’s readiness for the primetime
- Try Julia. Improve Julia :D

Discourse Thread

https://ebrmn.space/
https://jakegines.in/

We’re Applying to
PhD programs!!

</> https://github.com/EdwardBerman/SoJ/tree/main

Language Adoption

- The momentum idea is **incomplete**
- Programming languages like Rust have emerged, despite other languages having (potentially) similar utility

Julia’s lack of maturity

- Debugging

Julia’s lack of maturity

- Debugging
    - Long, unparsable stack traces

Julia’s lack of maturity

- Debugging
  - Long, unparsable stack traces
- Precompile Times and Executables

Julia’s lack of maturity

- Debugging
  - Long, unparsable stack traces
- Precompile Times and Executables
  - Historical technical debt

Language Adoption

- Julia lacks the following to truly emerge:
    - Industry Support

Language Adoption

- Julia lacks the following to truly emerge:
  - Industry Support
  - Strong testing infrastructure and software engineering features

Language Adoption

- Julia lacks the following to truly emerge:
    - Industry Support
    - Strong testing infrastructure and software engineering features
    - Interoperability (especially with C)

Language Adoption

- Julia lacks the following to truly emerge:
    - Industry Support
    - Strong testing infrastructure and software engineering features
    - Interoperability (especially with C)
    - A good debugger

Language Adoption

- Julia lacks the following to truly emerge:
  - Industry Support
  - Strong testing infrastructure and software engineering features
  - Interoperability (especially with C)
  - A good debugger
  - Precompilation

Julia’s lack of maturity

- Debugging
  - Long, unparsable stack traces
- Precompile Times and Executables
  - Historical technical debt
  - Not optimized for kernels and small exes like C/C++ [J.B. 2024 Julia Dispatch]