QM@pt > network

CF-UM-UP

Universidade do Minho
Escola de Ciências

Introduction to the julia programming language

Bruno Amorim

Quantum Agora, 25th March 2021

Disclaimer:

I am a bad programmer (in python,C and Fortran)
I am slightly better with Julia

Why do we need another programming language?

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! ☹

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! ☹

☺ **Fast** languages:

- Fortran
- C
- C++
- ...

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! ☹

☺ **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! ☹

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! ☹

☺ **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! ☹

Compromise:
1) Prototype in easy language
2) When performance becomes critical, rewrite parts/whole code in fast language

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematica
- Matlab
- ...

But slow! ☹

☺ **Fast** languages:

- Fortran
- C
- C++
- ...

But hard! ☹

Compromise:
1) Prototype in easy language
2) When performance becomes critical, rewrite parts/whole code in fast language

Problems:
- Duplication of effort
- Need to learn two languages or fast parts become a blackbox!

Why do we need another programming language?

☺ **Easy** languages to write code:

- Python
- Mathematic
- Matlab
- ...

But slow!

☺ **Fast** languages:

Two language problem!

Compromise
1) Prototype i
2) When perfo fast language

Problems:
- Duplication of effort
- Need to learn two languages or fast parts become a blackbox!

Enter Julia

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman

In short, because we are greedy. ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

Enter Julia

From https://julialang.org/blog/2012/02/why-we-created-julia/

“ **Why We Created Julia**

14 February 2012 | **Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman**

Jeff Bezanson Stefan Karpinski Viral B. Shah Alan Edelman ”

In short, because we are greedy.

“ We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

(Did we mention it should be as fast as C?) ”

So what is Julia?

So what is Julia?

New language: 1st public realease in 2012, stable 1.0 version in 2018
Current version: 1.6.0 (released yesterday!)
Initially development at MIT:

- *Julia: A Fast Dynamic Language for Technical Computing,* ***arXiv:1209.5145***
  Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman
- *Julia: A Fresh Approach to Numerical Computing,* **arXiv:1411.1607**
  Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
- *Abstraction in technical computing* (PhD Thesis)
  https://github.com/JeffBezanson/phdthesis/blob/master/main.pdf
  Jeff Bezanson

So what is Julia?

New language: 1st public realease in 2012, stable 1.0 version in 2018
Current version: 1.6.0 (released yesterday!)
Initially development at MIT:

- *Julia: A Fast Dynamic Language for Technical Computing,* ***arXiv:1209.5145***
  Jeff Bezanson, Stefan Karpinski, Viral B. Shah, Alan Edelman
- *Julia: A Fresh Approach to Numerical Computing,* **arXiv:1411.1607**
  Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
- *Abstraction in technical computing* (PhD Thesis)
  https://github.com/JeffBezanson/phdthesis/blob/master/main.pdf
  Jeff Bezanson

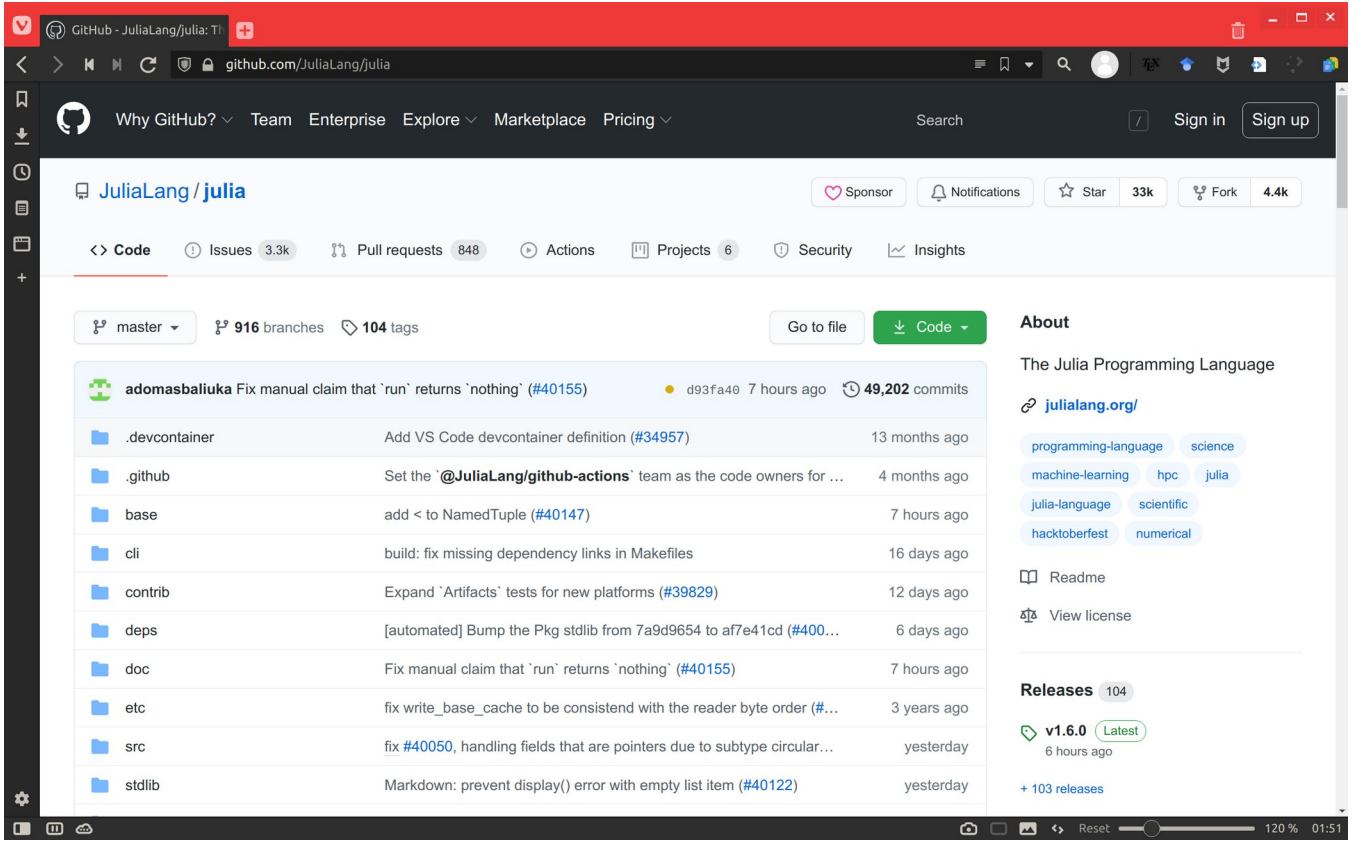
- Open source
- Solves the two language problem
- Interactive
- High performance
- Simple syntax
- Aimed at scientific computing

Open source

source code hosted on github

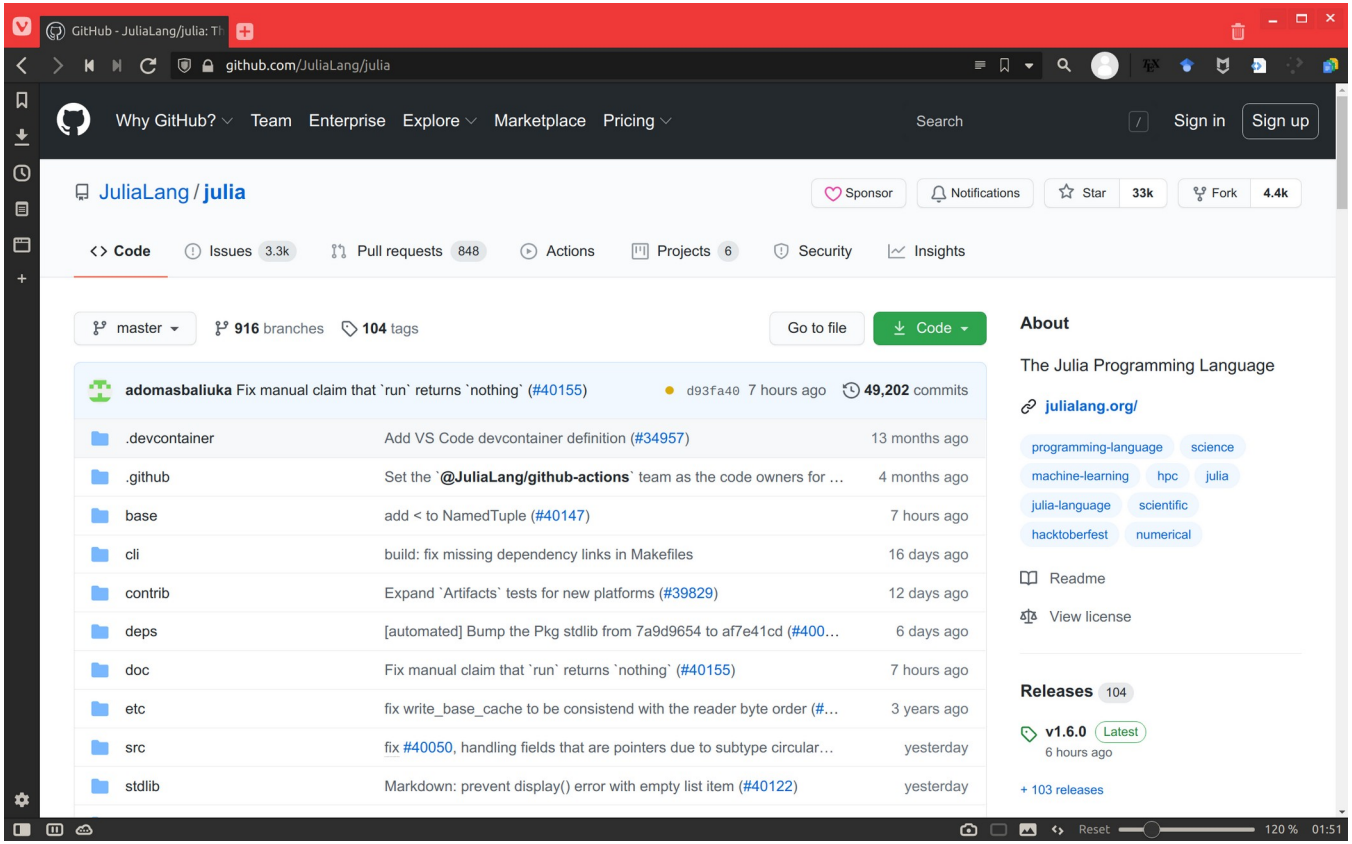
Open source

source code hosted on github

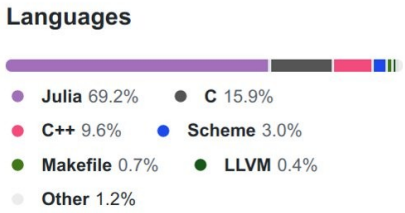


Open source

source code hosted on github



Many contributers



Julia is mostly written in Julia

This is true for most packages!

Solving the two language problem

Julia packages are mostly written in Julia

Solving the two language problem

Julia packages are mostly written in Julia

JuliaLinearAlgebra / **IterativeSolvers.jl**

Code | Issues 40 | Pull requests 5 | Actions | Projects | Security | Insights

Languages

Julia 100.0%

SciML / **DifferentialEquations.jl**

Code | Issues 110 | Pull requests | Actions | Projects | Security | Insights

Languages

Julia 96.4% **TeX** 3.6%

Solving the two language problem

Julia packages are mostly written in Julia

JuliaLinearAlgebra / **IterativeSolvers.jl**

Code | Issues 40 | Pull requests 5 | Actions | Projects | Security | Insights

Languages

Julia 100.0%

SciML / **DifferentialEquations.jl**

Code | Issues 110 | Pull requests | Actions | Projects | Security | Insights

Languages

Julia 96.4% **TeX** 3.6%

contrast with

numpy / **numpy**

Code | Issues 2k | Pull requests 235 | Actions | Projects 8 | Wiki | Security | Insights

Languages

Python 63.5% **C** 35.2%
C++ 1.0% **JavaScript** 0.1%
Fortran 0.1% **Shell** 0.1%

scipy / **scipy**

Code | Issues 1.3k | Pull requests 319 | Actions | Projects | Wiki | Security | Insights

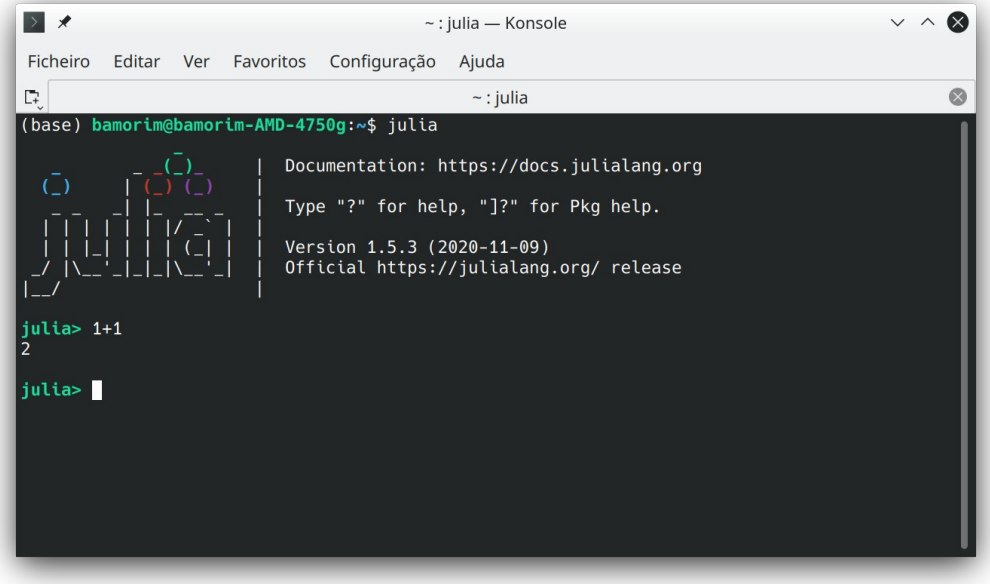
Languages

Python 51.8% **Fortran** 19.2%
C 17.3% **C++** 11.4%
TeX 0.2% **CMake** 0.1%

Julia is an interactive language

Julia is an interactive language

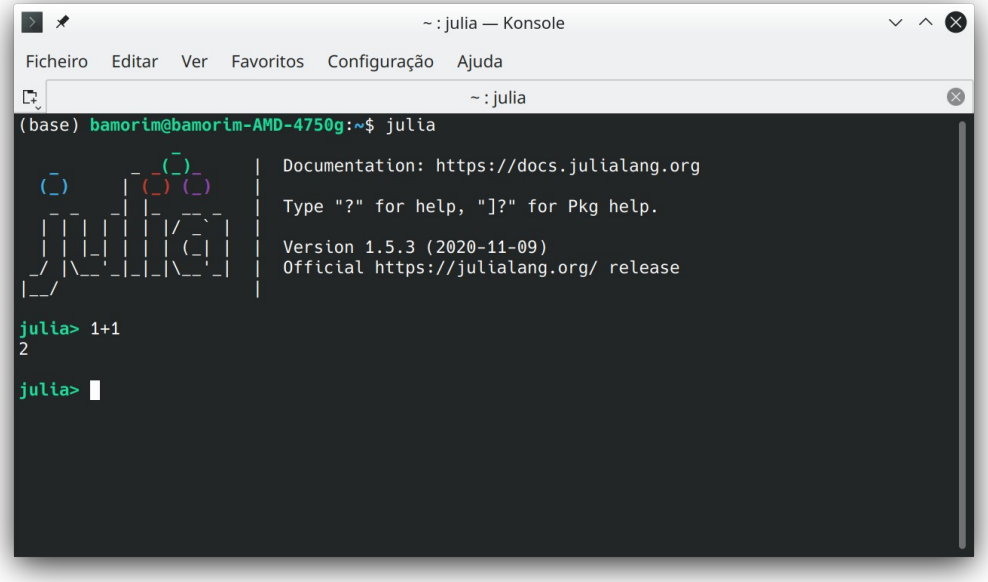
REPL (Read Eval Print Loop shell)

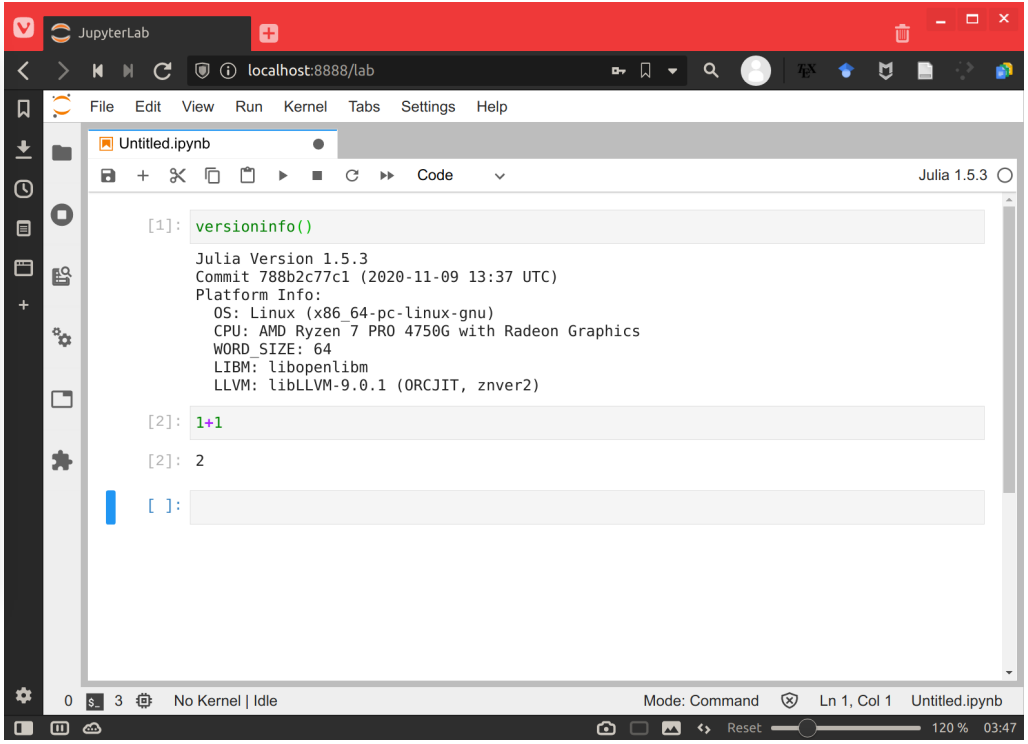


Julia is an interactive language

REPL (Read Eval Print Loop shell)

jupyter notebooks



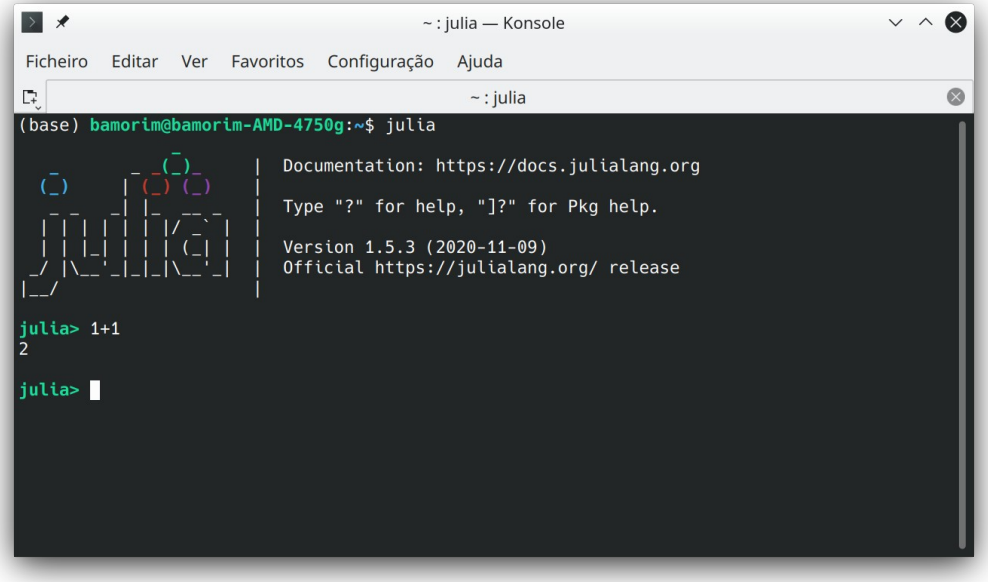


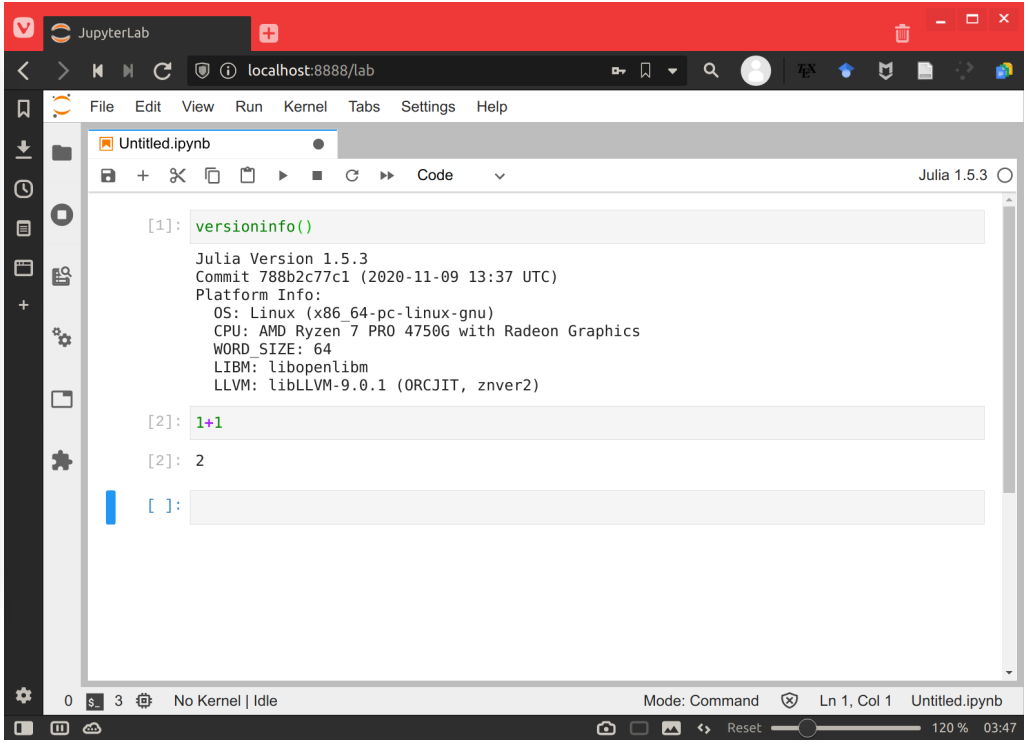
Julia is an interactive language

REPL (Read Eval Print Loop shell)

jupyter notebooks

Julia

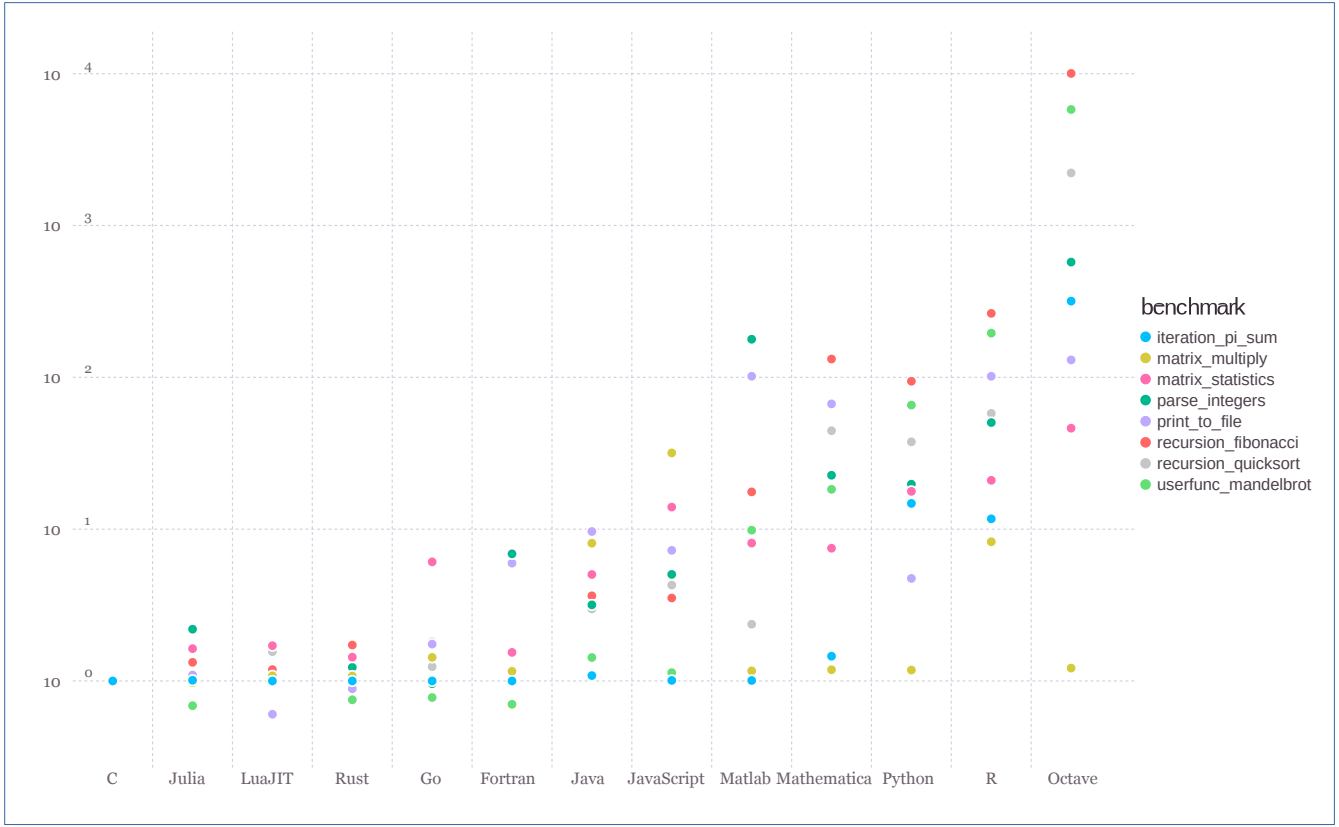




Julia is a high performance language

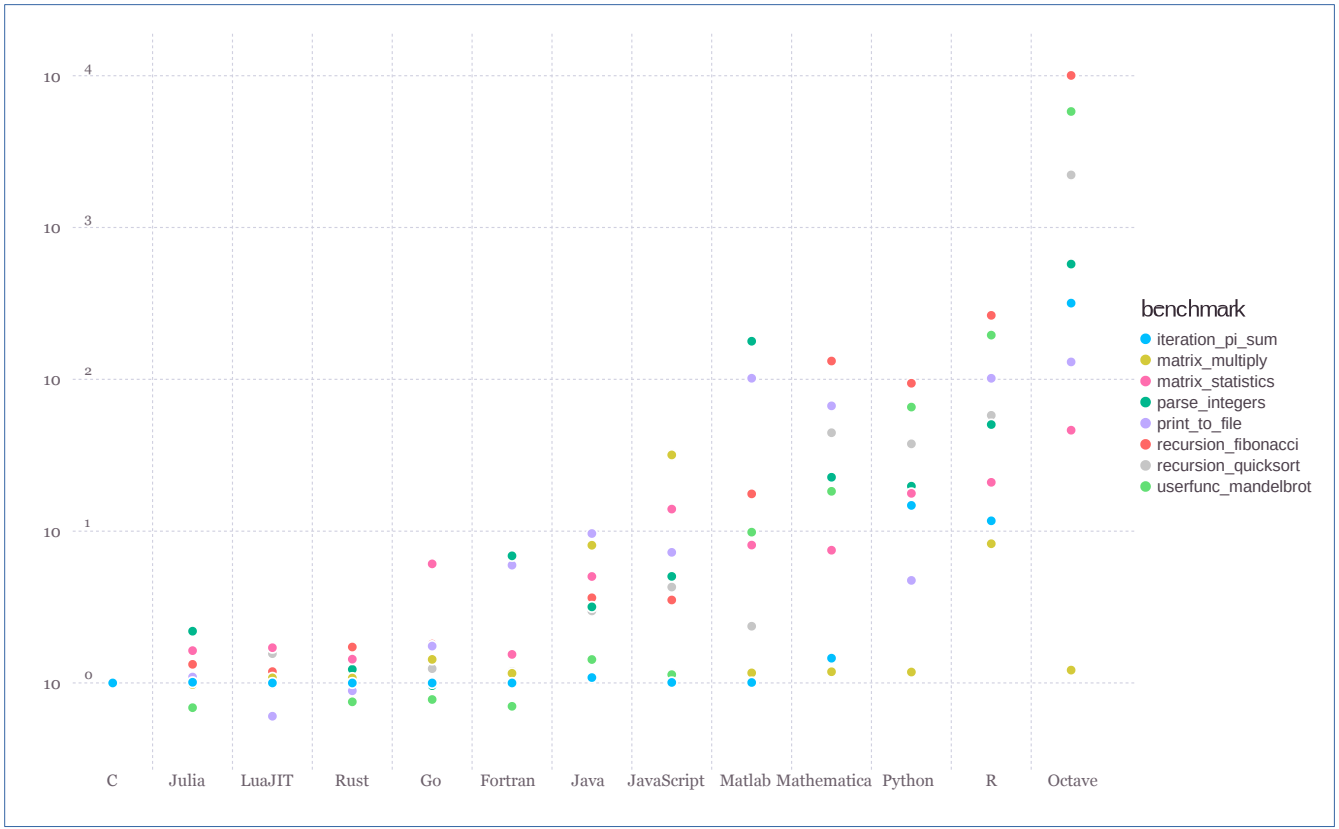
Julia is a high performance language

Julia microbenchmarks from https://julialang.org/benchmarks/



Julia is a high performance language

Julia microbenchmarks from https://julialang.org/benchmarks/



Julia is actually a compiled language!

It uses LLVM (same backend as the Clang C compiler)

Julia has nice syntax

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of 2pi/6:

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of 2pi/6:

Python code:

```
import numpy as np

def c6rotation(n):

    return np.array([
        [np.cos(2*np.pi*n/6), -np.sin(2*np.pi*n/6)],
        [np.sin(2*np.pi*n/6), np.cos(2*np.pi*n/6)]
    ])
```

Julia has nice syntax

Let us define a function that returns a rotation by an angle that is a multiple of 2pi/6:

Julia code:

```julia
c6rotation(n) = [
    cos(2π*n/6) -sin(2π*n/6);
    sin(2π*n/6) cos(2π*n/6)
]
```

Python code:

```python
import numpy as np

def c6rotation(n):

    return np.array([
        [np.cos(2*np.pi*n/6), -np.sin(2*np.pi*n/6)],
        [np.sin(2*np.pi*n/6), np.cos(2*np.pi*n/6)]
    ])
```

Maybe this sparked your interest

But you might be asking:

“Is anyone using Julia?”

Julia adoption

Julia adoption

Julia adoption evolution between 1st Jan 2020 and 1st Jan 2021:

	Total Cumulative as of Jan 1, 2020	Total Cumulative as of Jan 1, 2021	Change
Number of Julia Downloads (JuliaLang + Docker + JuliaPro)	12,950,630	24,205,141	+87%
Number of Julia Packages	2,787	4,809	+73%
GitHub stars (Julia language repo + registered packages)	99,830	161,774	+62%
YouTube views (Julia language channel)	1,562,223	3,320,915	+113%
Published citations of Julia: A Fast Dynamic Language for Technical Computing (2012) + Julia: A Fresh Approach to Numerical Computing (2017)	1,680	2,531	+51%
Discourse posts	137,399	211,888	+54%
TIOBE Index Rank	#47	#23	+24

from: https://juliacomputing.com/blog/2021/01/newsletter-january/

Julia adoption

Julia adoption evolution between 1st Jan 2020 and 1st Jan 2021:

	Total Cumulative as of Jan 1, 2020	Total Cumulative as of Jan 1, 2021	Change
Number of Julia Downloads (JuliaLang + Docker + JuliaPro)	12,950,630	24,205,141	+87%
Number of Julia Packages	2,787	4,809	+73%
GitHub stars (Julia language repo + registered packages)	99,830	161,774	+62%
YouTube views (Julia language channel)	1,562,223	3,320,915	+113%
Published citations of Julia: A Fast Dynamic Language for Technical Computing (2012) + Julia: A Fresh Approach to Numerical Computing (2017)	1,680	2,531	+51%
Discourse posts	137,399	211,888	+54%
TIOBE Index Rank	#47	#23	+24

from: https://juliacomputing.com/blog/2021/01/newsletter-january/

Companies using Julia

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

intel
Disney
BNDES
O BANCO DO DESENVOLVIMENTO DE TODOS OS BRASILEIROS
amazon
Capital One
KPMG
Google
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Microsoft
IBM
BLACKROCK
NASA
Ford
OAK RIDGE
National Laboratory
AstraZeneca
FEDERAL AVIATION ADMINISTRATION
AVIVA
WESTERN ASSET
abbvie
Chevron
ExxonMobil
FEDERAL RESERVE BANK OF NEW YORK
INCORPORATED MAY 18, 1914
gsk
JPMORGAN CHASE & CO.
Pfizer
moderna
messenger therapeutics
SANOFI
STATE STREET
Takeda
EY
Building a better working world

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Science Labs

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Tech

intel
Disney
BNDES
O BANCO DO DESENVOLVIMENTO DE TODOS OS BRASILEIROS
amazon
Capital One
KPMG
Google
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Microsoft
IBM
BLACKROCK
NASA
Ford
OAK RIDGE National Laboratory
AstraZeneca
FEDERAL AVIATION ADMINISTRATION
AVIVA
WESTERN ASSET
abbvie
Chevron
ExxonMobil
FEDERAL RESERVE BANK OF NEW YORK · INCORPORATED MAY 18, 1914 ·
gsk
JPMORGAN CHASE & CO.
Pfizer
moderna
messenger therapeutics
SANOFI
STATE STREET
Takeda
EY Building a better working world

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Finnance/Banking/Assurance

intel
Disney
BNDES
O BANCO DO DESENVOLVIMENTO DE TODOS OS BRASILEIROS
amazon
Capital One
KPMG
Google
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Microsoft
IBM
BLACKROCK
NASA
Ford
OAK RIDGE National Laboratory
AstraZeneca
FEDERAL AVIATION ADMINISTRATION
AVIVA
WESTERN ASSET
abbvie
Chevron
ExxonMobil
FEDERAL RESERVE BANK OF NEW YORK · INCORPORATED MAY 18, 1914 ·
gsk
JPMORGAN CHASE & CO.
Pfizer
moderna
messenger therapeutics
SANOFI
STATE STREET
Takeda
EY Building a better working world

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

Pharma

intel
Disney
BNDES
O BANCO DO DESENVOLVIMENTO DE TODOS OS BRASILEIROS
amazon
Capital One
KPMG
Google
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Microsoft
IBM
BLACKROCK
NASA
Ford
OAK RIDGE National Laboratory
AstraZeneca
FEDERAL AVIATION ADMINISTRATION
AVIVA
WESTERN ASSET
abbvie
Chevron
ExxonMobil
FEDERAL RESERVE BANK OF NEW YORK · INCORPORATED MAY 18, 1914 ·
gsk
JPMORGAN CHASE & CO.
Pfizer
moderna
messenger therapeutics
SANOFI
STATE STREET
Takeda
EY Building a better working world

Companies using Julia

From https://juliacomputing.com:

JULIA USERS AND JULIA COMPUTING CUSTOMERS

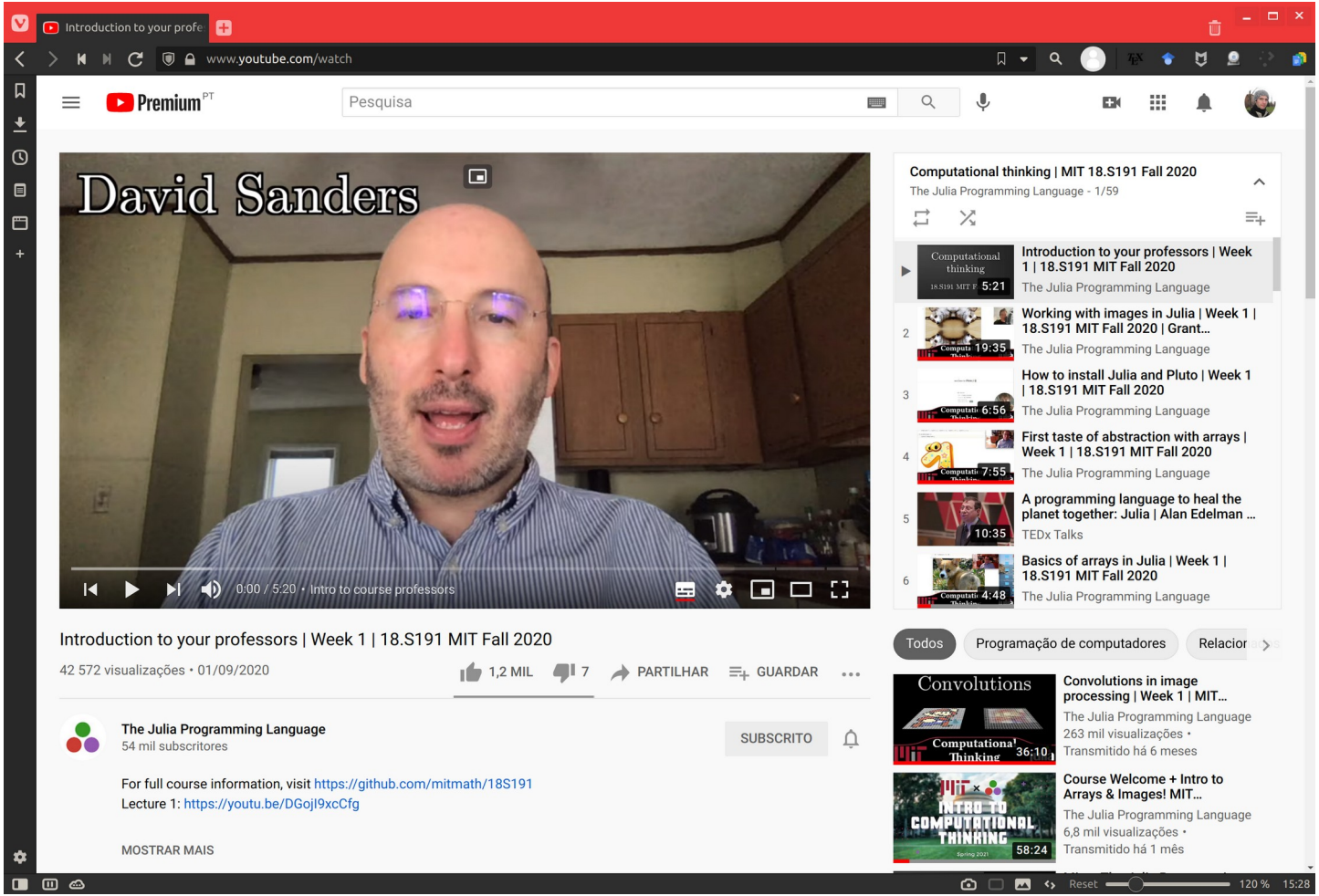
Covid?

intel
Disney
BNDES
O BANCO DO DESENVOLVIMENTO DE TODOS OS BRASILEIROS
amazon
Capital One
KPMG
Google
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Microsoft
IBM
BLACKROCK
NASA
Ford
OAK RIDGE National Laboratory
AstraZeneca
FEDERAL AVIATION ADMINISTRATION
AVIVA
WESTERN ASSET
abbvie
Chevron
ExxonMobil
FEDERAL RESERVE BANK OF NEW YORK · INCORPORATED MAY 18, 1914 ·
gsk
JPMORGAN CHASE & CO.
Pfizer
moderna
messenger therapeutics
SANOFI
STATE STREET
Takeda
EY Building a better working world

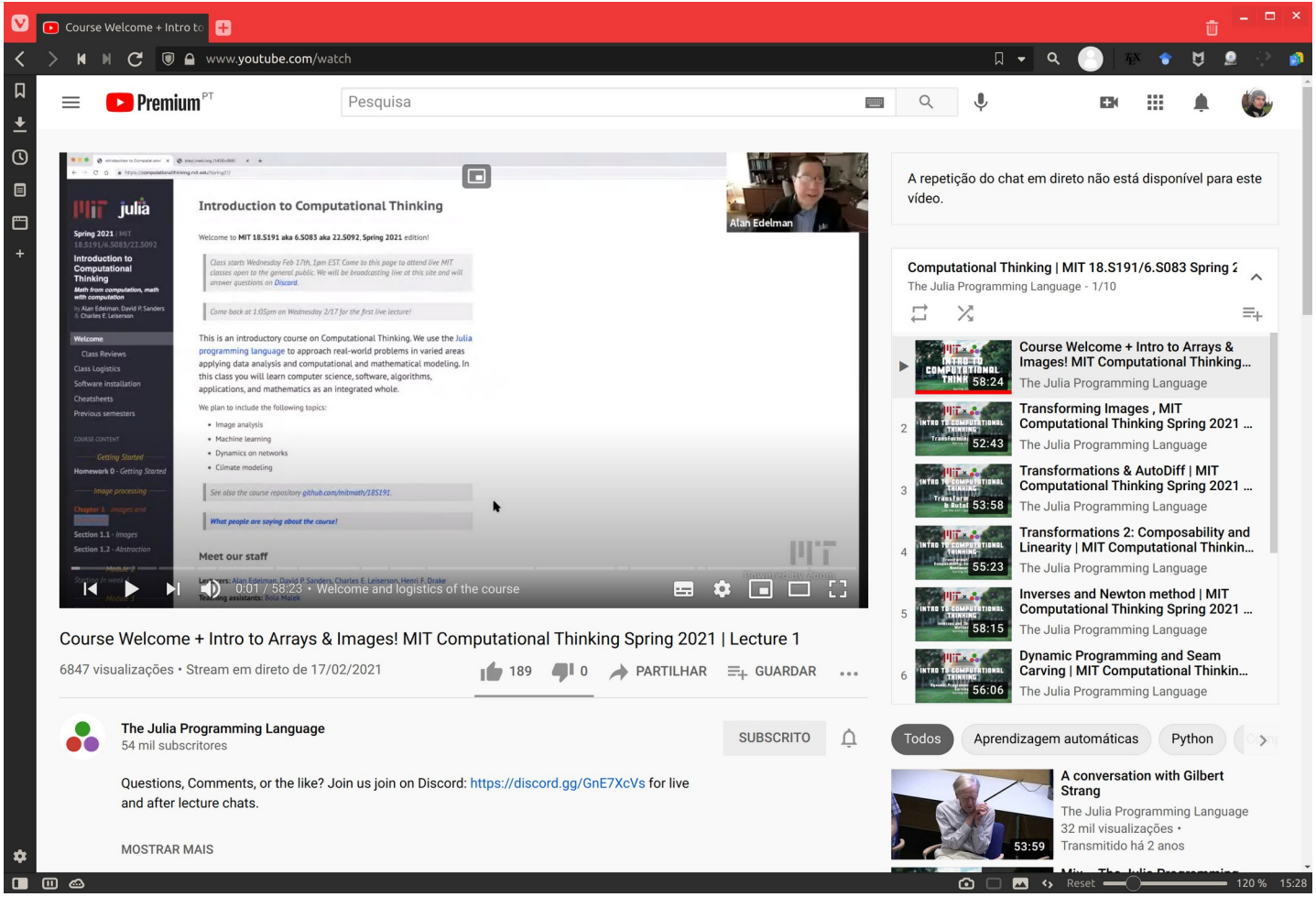
Most importantly:

Julia is being taugh at universities!

Julia course at MIT



Julia course at MIT



Julia course at MIT

It is being taught for some time:

- Massachusetts Institute of Technology (MIT)
  - 6.251 / 15.081, Introduction to Mathematical Programming (Prof. Dimitris J. Bertsimas), Fall 2015
  - 18.06, Linear Algebra: Fall 2015, Dr. Alex Townsend; Fall 2014, Prof. Alexander Postnikov; Fall 2013, Prof. Alan Edelman
  - 18.303, Linear Partial Differential Equations: Analysis and Numerics (Prof. Steven G. Johnson), Fall 2013–2016.
  - 18.337 / 6.338, Numerical Computing with Julia (Prof. Alan Edelman). Fall 2015 (IJulia notebooks). Fall 2013–
  - 18.085 / 0851, Computational Science And Engineering I (Prof. Pedro J. Sáenz)
  - 18.330, Introduction to Numerical Analysis (Dr. Homer Reid), Spring 2013–2015
  - 18.335, Introduction to Numerical Methods (Prof. Steven G. Johnson), Fall 2013, Spring 2015
  - 18.338, Eigenvalues Of Random Matrices (Prof. Alan Edelman), Spring 2015
  - 18.S096, Performance Computing in a High Level Language (Steven G. Johnson, Alan Edelman, David Sanders, Jeff Bezanson), January 2017.
  - 15.093 / 6.255, Optimization Methods (Prof. Dimitris Bertsimas and Dr. Phebe Vayanos), Fall 2014
  - 15.S60, Software Tools for Operations Research (Iain Dunning), Spring 2014
  - 15.083, Integer Programming and Combinatorial Optimization (Prof. Juan Pablo Vielma), Spring 2014

From: https://julialang.org/learning/classes/

Julia in other universities:

From: https://julialang.org/learning/

For a more detailed list: https://julialang.org/learning/classes/

Projects using Julia: Celeste project

Projects using Julia: Celeste project

Project to catalog Sloan Digital Sky Survey

- processed ~ 178 terabytes of data in 14.6 minutes
- peak performance: 1.54 **petaflops**
- used **1.3 million threads** running on **9 300 nodes** at NERSC (National Energy Research Scientific Computing Center)
- written **entirely in Julia**

Projects using Julia: Celeste project

Project to catalog Sloan Digital Sky Survey

- processed ~ 178 terabytes of data in 14.6 minutes
- peak performance: 1.54 **petaflops**
- used **1.3 million threads** running on **9 300 nodes** at NERSC (National Energy Research Scientific Computing Center)
- written **entirely in Julia**

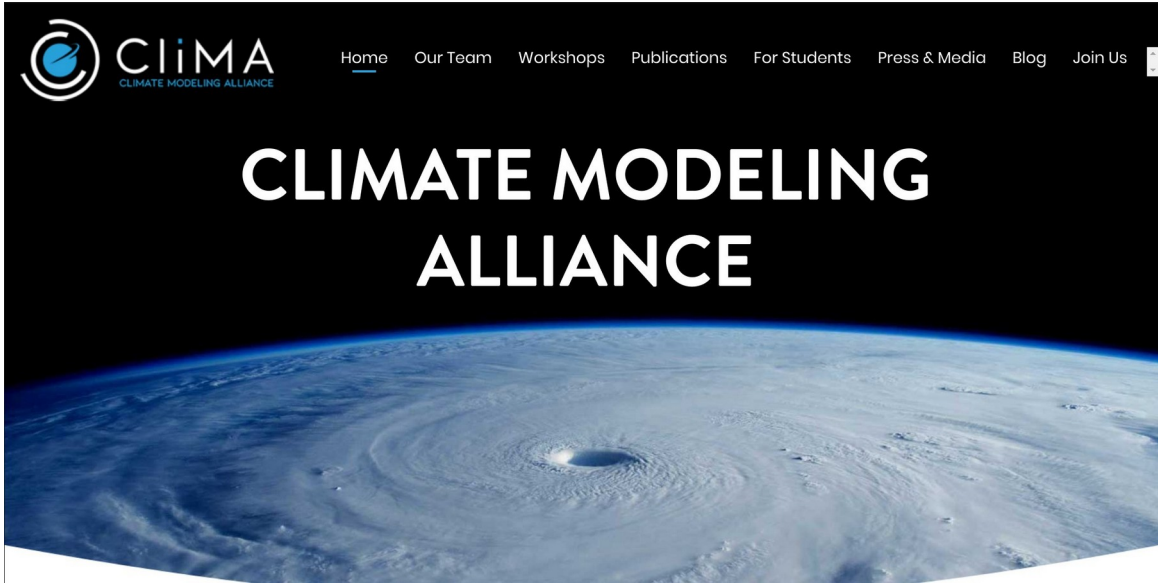
For more info:
- https://juliacomputing.com/case-studies/celeste/
- https://youtu.be/uecdcADM3hY
- https://github.com/jeff-regier/Celeste.jl

Projects using Julia: Clima project

Projects using Julia: Clima project

https://clima.caltech.edu

New climate model built from the ground up in Julia
Research consurtium:
- Caltech
- MIT
- Naval Postgraduate School
- Jet Propulsion Laboratory



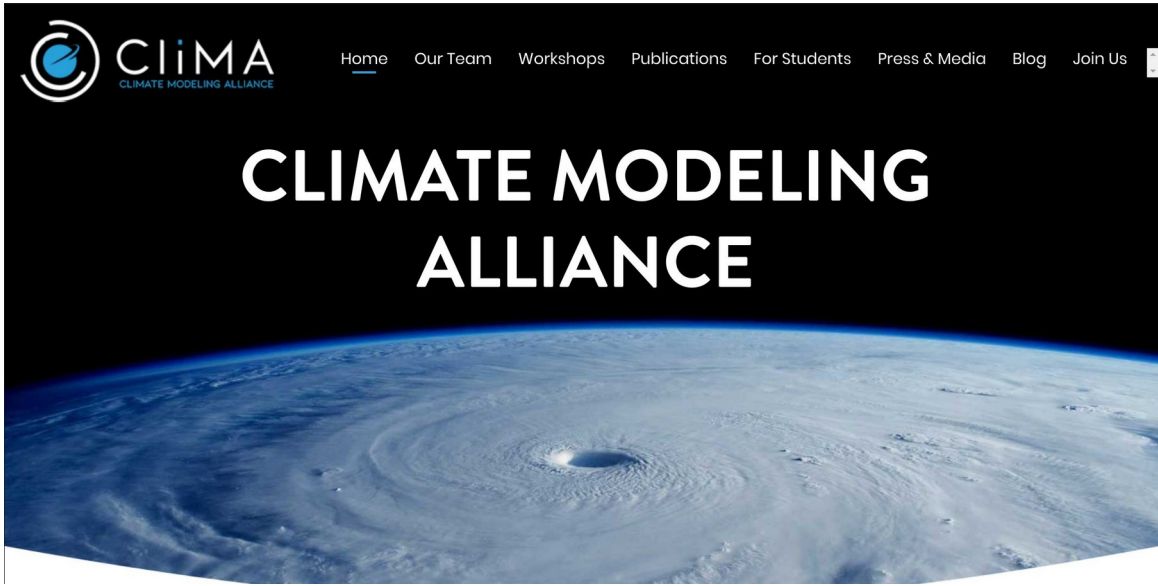
Projects using Julia: Clima project

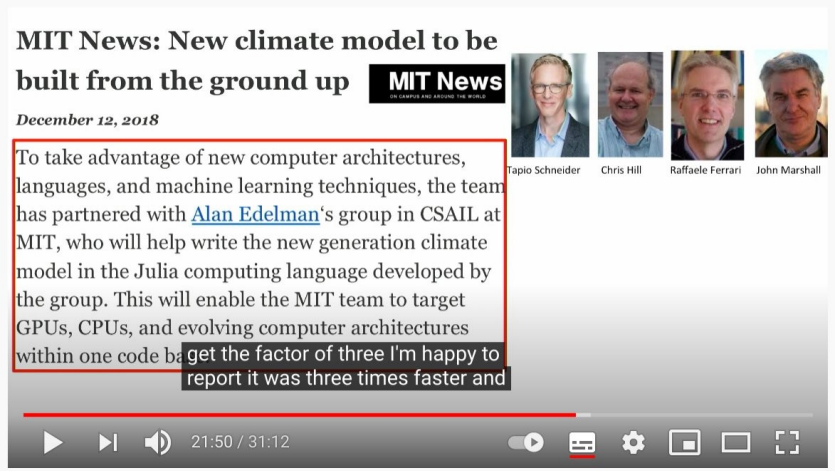
New climate model built from the ground up in Julia

https://clima.caltech.edu

Research consurtium:

- Caltech
- MIT
- Naval Postgraduate School
- Jet Propulsion Laboratory





Where OK with a 3x slowdown.
Ended up being 3x faster!

https://youtu.be/nwdGsz4rc3Q?t=1238

https://www.hpcwire.com/2020/01/14/julia-programmings-dramatic-rise-in-hpc-and-elsewhere/

Rich package ecosystem

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, Vegalite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, Vegalite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Physics related packages:

- **Tensor Networks:** Itensors.jl
- **Quantum Computing:** Yao.jl
- **Quantum optics:** QuantumOptics.jl
- **DFT:** DFTK.jl
- **Quantum Monte Carlo:** MonteCarlo.jl
- **Tight-binding:** Quantica.jl

Rich package ecosystem

- **Iterative Linear Solvers:** IterativeSolvers.jl, KrylovKit.jl
- **Eigensolvers:** ArnoldiMethod.jl, KrylovKit.jl
- **Differential equations:** DifferentialEquations.jl
- **Integration:** QuadGK.jl, FastGaussQuadrature.jl, Quadrature.jl
- **Optimization:** Optim.jl, JuMP.jl
- **Plotting:** Plots.jl, Pyplot.jl, Vegalite.jl, Makie.jl, Gadfly.jl
- **Machine learning:** Flux.jl, Knet.jl

Physics related packages:

- **Tensor Networks:** Itensors.jl
- **Quantum Computing:** Yao.jl
- **Quantum optics:** QuantumOptics.jl
- **DFT:** DFTK.jl
- **Quantum Monte Carlo:** MonteCarlo.jl
- **Tight-binding:** Quantica.jl

You can search for more using:
- juliahub.com/ui/Packages
- juliaobserver.com/packages
- just use google

More resources to learn about julia

- **Julia page:** julialang.org
- **Julia documentation:** docs.julialang.org
- **Julia forum:** discourse.julialang.org
- **Julia youtube channel:** www.youtube.com/user/JuliaLanguage/playlists

More tutorials and books are listed in julialang.org/learning