Operations research with Julia and JuMP

Pedro Belin Castellucci

February, 2017

Online material at github.com/pedrocastellucci/athena.

Mathematical programming

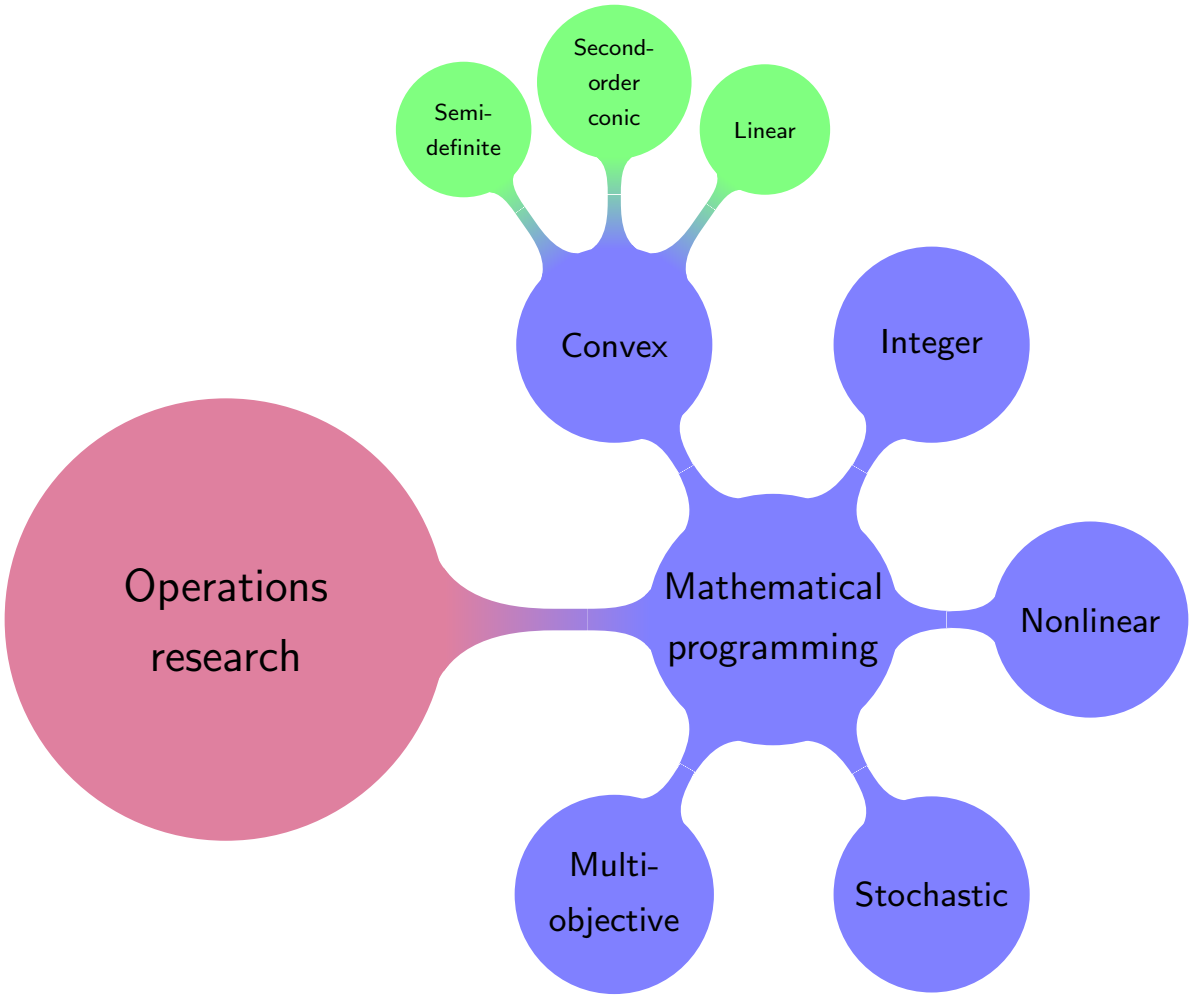

Figure 1: Information from `en.wikipedia.org/wiki/Mathematical_optimization`.

Which can JuMP handle?

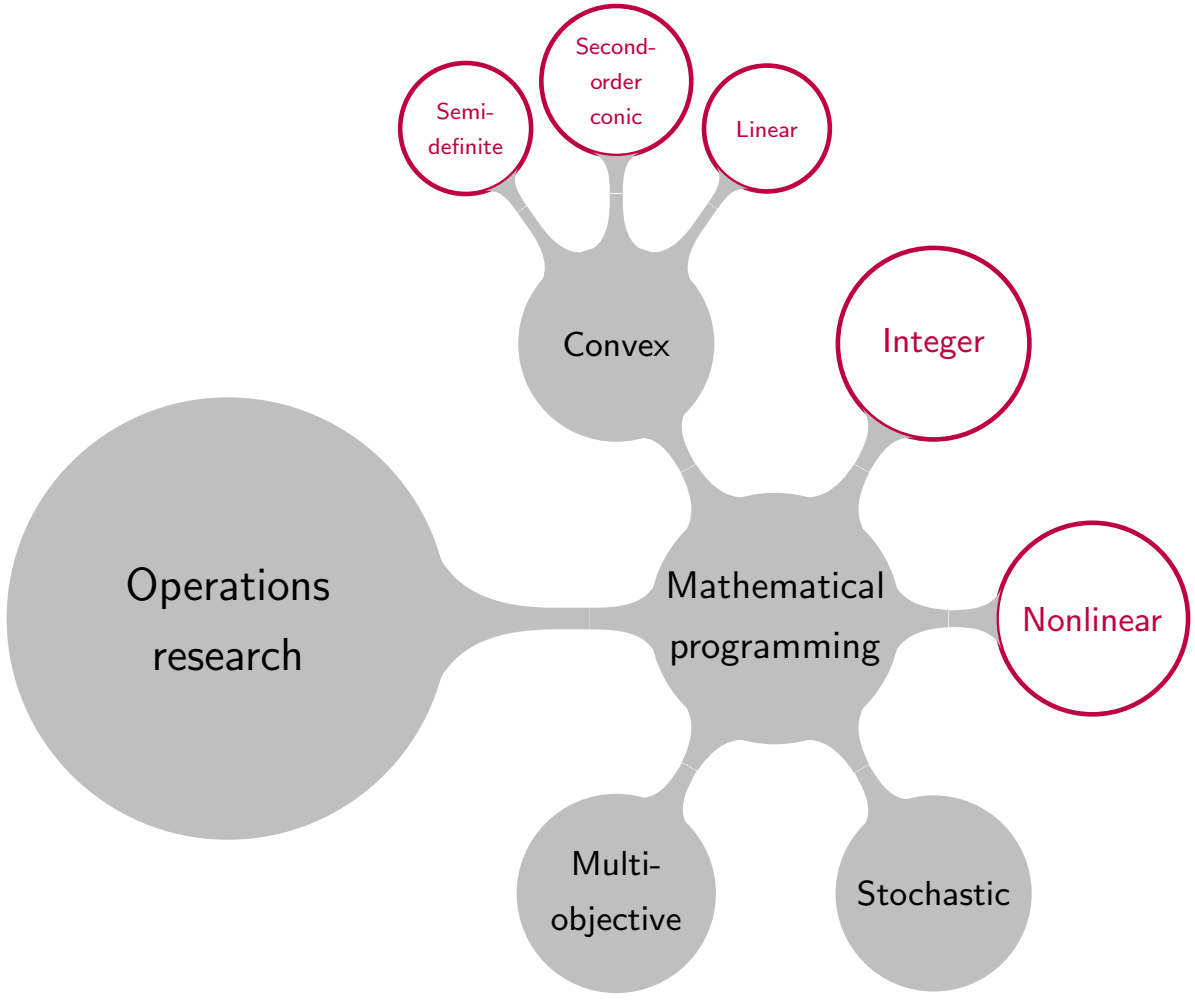

Figure 2: For JuMP documentation check `www.juliaopt.org/JuMP.jl/0.15/`.

Introduction to Julia programming

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))
julialang.org/blog/2012/02/why-we-created-julia

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

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

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

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

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

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

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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.

"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

"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?)"

(Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (2012))

`julialang.org/blog/2012/02/why-we-created-julia`

How fast is it?

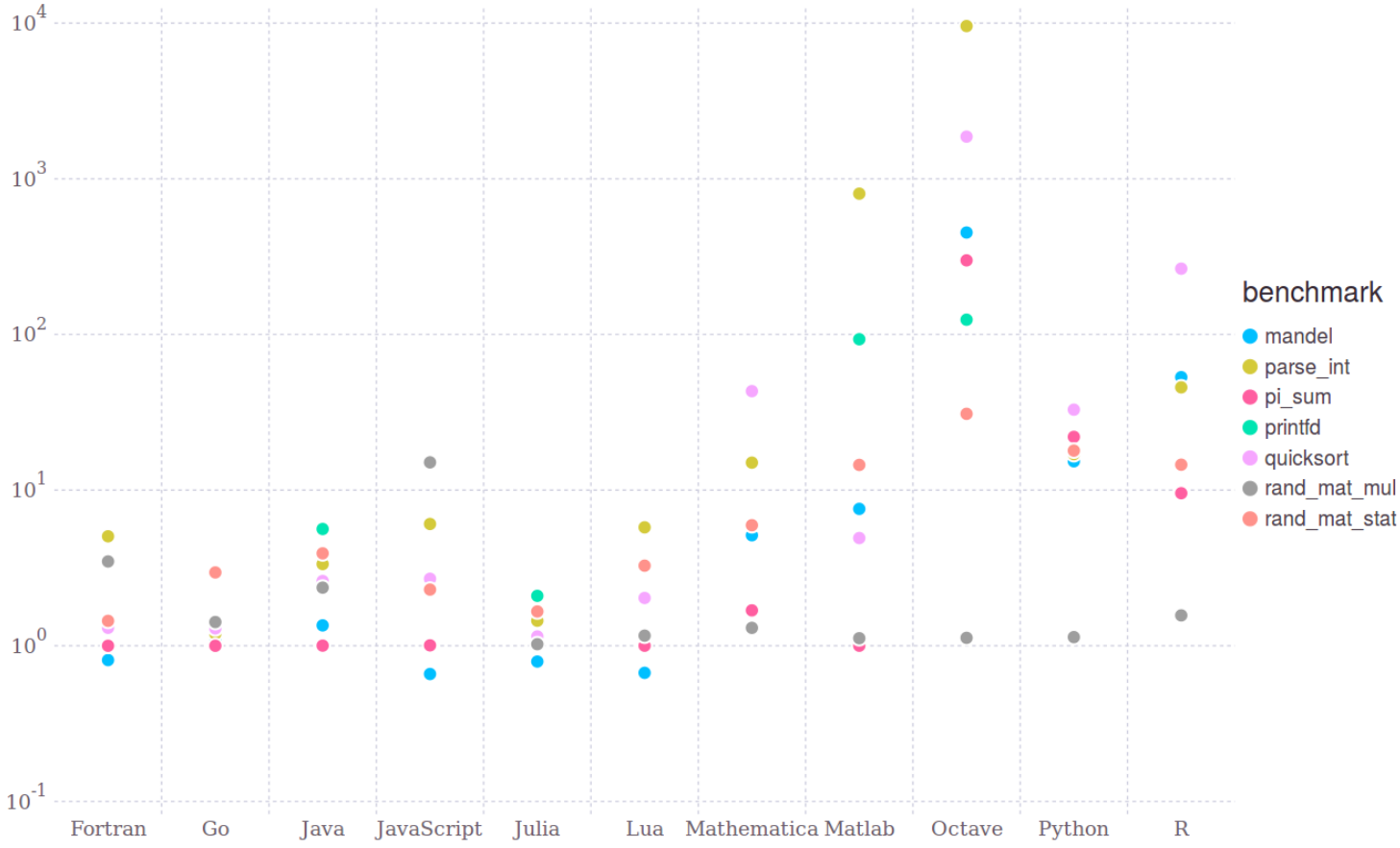

Figure 3: Speed relative to C smaller is better (C performance is 1.0). According to `julialang.org/benchmarks/`.

Who is using Julia?

Stanford University. Introduction to Multidisciplinary Design Optimization (Prof. Mykel J. Kochenderfer).

MIT. Integer Programming and Combinatorial Optimization (Prof. Juan Pablo Vielma).

MIT. Optimization Methods (Prof. Dimitris Bertsimas and Dr. Phebe Vayanos).

University at Buffalo. Linear Programming (Prof. Changhyun Kwon).

"Sapienza" University of Rome. Operations Research (Giampaolo Liuzzi).

University of South Florida. Nonlinear Optimization and Game Theory (Prof. Changhyun Kwon).

For more check http://julialang.org/teaching/.

What can we do with Julia?

Simple Audio IO in Julia	(AudioIO),
A neural network	(BackpropNeuralNet),
Support vector machines	(LIBSVM, LIBLINEAR),
Machine learning	(MachineLearning),
Bioinformatics and Computational Biology	(Bio),
Curve fitting	(CurveFit),
Describe and model financial markets	(FinancialMarkets),
Black-box optimization	(BlackBoxOptim),
Combinatorics	(Combinatorics),
Evolutionary and genetic algorithms	(Evolutionary),

Gurobi, GLPK, CPLEX, Cbc, Clp CoinOptServices, JuMP.

For more check: http://pkg.julialang.org/

What will we do?

Basic Julia programming.

Explore JuMP for mixed integer linear problems.

Lesson one

Arrays are indexed starting at one.

[1] www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html

Lesson one

Arrays are indexed starting at one.

Why numbering should start at zero?[1]

[1] www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html

Lesson one

Arrays are indexed starting at one.

Why numbering should start at zero?[1] "I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras." (Alan Kay)

[1] www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html

Installing Julia

On Linux/Ubuntu:

sudo add-apt-repository ppa:staticfloat/juliareleases

sudo apt-get update

sudo apt-get install julia

For other platforms check

julialang.org/downloads/platform.html.