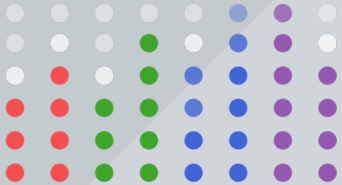


Julia in Pharma

Dr. Viral B. Shah (Julia Computing)

Dr. Vijay Ivaturi (Pumas-AI)

R/Pharma 2020

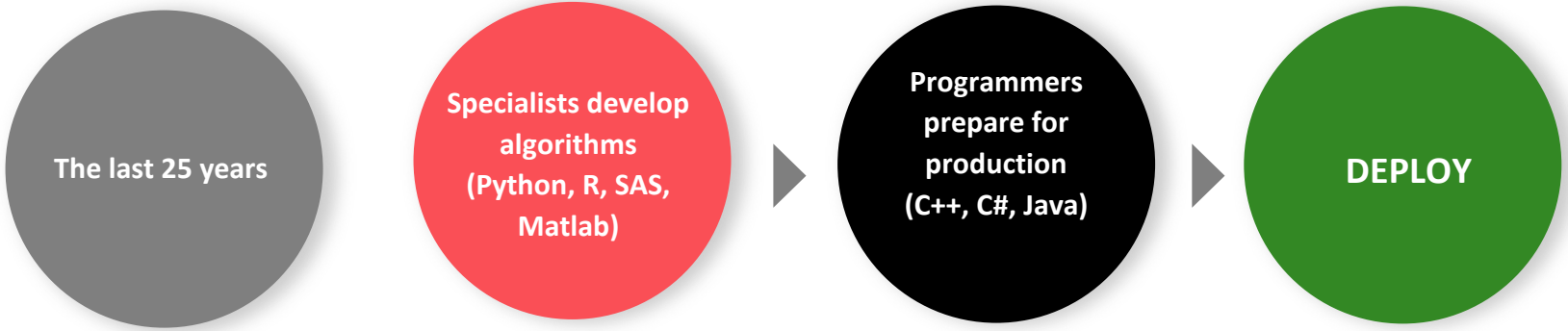
Julia computing

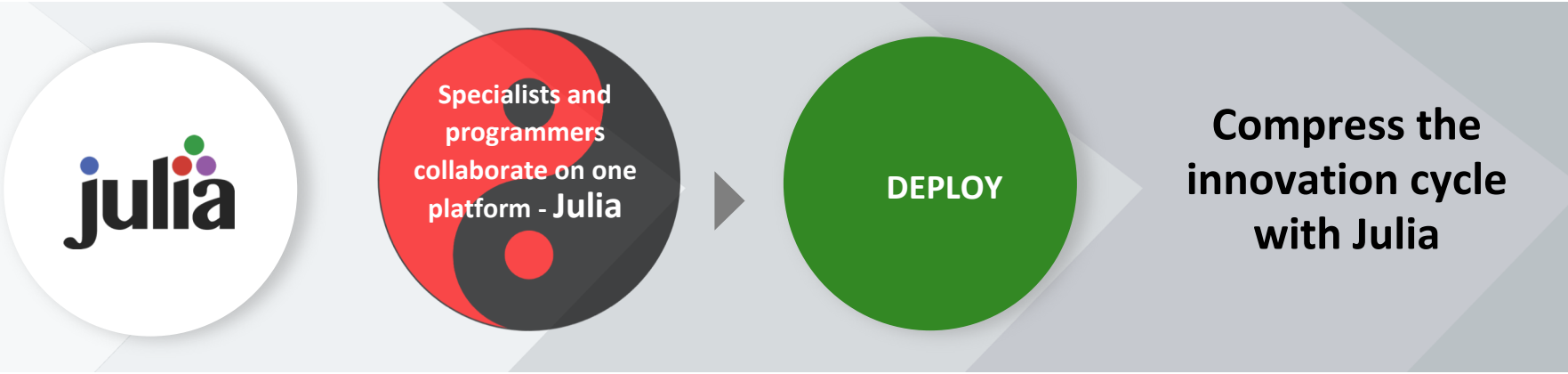
Email: info@juliacomputing.com

pumas AI

Email: info@pumas.ai

We created Julia to solve the two language problem





Origin and Journey

1.	2009	An email thread about a new language
2.	2012	Why we created Julia. John Myles White: Julia, I Love you Doug Bates: A Julia version of the multinomial sampler
3.	2013	Julia becomes the "Ju" in Jupyter JuliaStats: Rmath integrated. Distributions.jl. GLM.jl
4.	2015	Julia co-creators found Julia Computing, Inc. Doug Bates: Rcall: Running embedded R in Julia
5.	2017	1M Julia downloads
5.	2018	Julia 1.0.
6.	2019	10M Julia downloads. Wilkinson Prize. Sidney Fernbach Prize.
7.	2020	20M Julia downloads. Pumas.ai founded. Julia 1.5 released

Dr. Viral Shah Prof. Alan Edelman

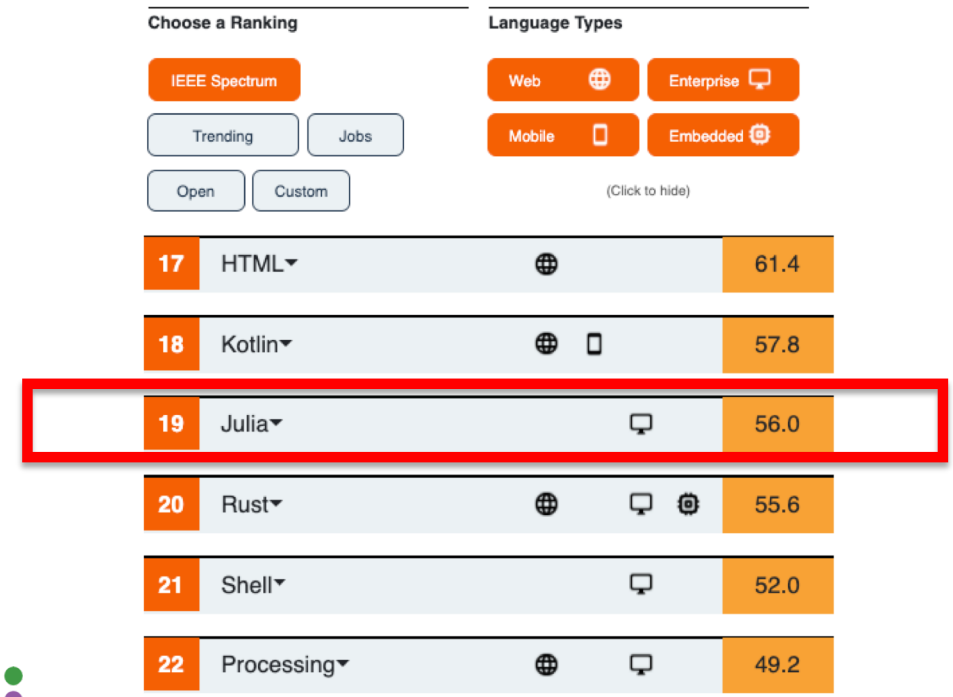
Dr. Jeff Bezanson Stefan Karpinski

Over 10,000 organizations are using Julia

Over 1,500 Universities are using and teaching Julia

Rapid adoption for a young language

20M downloads; 4,000 packages; 10,000 companies; 1,500 universities



TOOLBOX

JULIA: COME FOR THE SYNTAX, STAY FOR THE SPEED

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.

ILLUSTRATION BY THE PROJECT TWINS

BY JEFFREY M. PERKEL

When it comes to climate modelling, every computational second counts. Designed to account for air, land, sun and sea, and the complicated physics that links them, these models can run to millions of lines of code, which are executed on the world's most powerful computers. So when the coder-climatologists of the Climate Modeling Alliance (CliMA) — a coalition of US-based scientists, engineers and mathematicians — set out to build a model from the ground up, they opted for a language that could handle their needs. They opted for Julia.

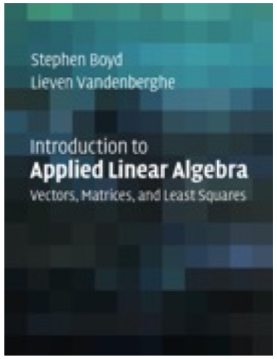
Launched in 2012, Julia is an open-source language that combines the interactivity and syntax of 'scripting' languages, such as Python, Matlab and R, with the speed of 'compiled' languages such as Fortran and C.

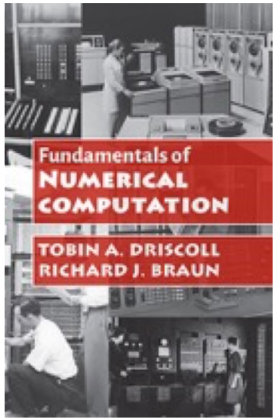
Among climate scientists, the lingua franca is Fortran: speedy, but — with roots dating to the 1950s — not terribly exciting. "A lot of people, when they hear 'Fortran,' are like, 'Oh, my God, I don't want to program in that,'" says Frank Giraldo, a mathematician at the Naval Postgraduate School in Monterey, California, and a co-principal investigator on the CliMA project. Younger programmers prefer languages that can accommodate the latest trends in software and hardware design, Giraldo says, and since adopting Julia he has seen an uptick in interest. "Some of them are really interested in climate modelling, but others are intrigued by the idea of using Julia for some large-scale application," he says.

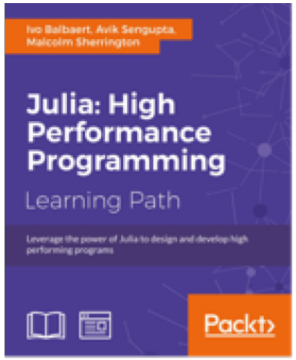
Jane Herriman, who is studying materials science at the California Institute of Technology in Pasadena, says that she has seen ten-

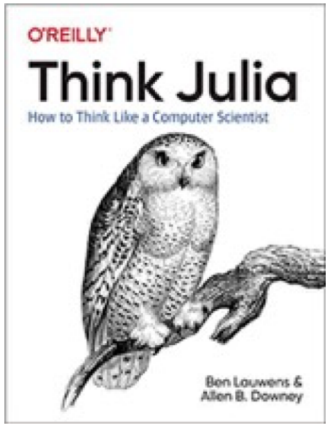
nature
International journal of science

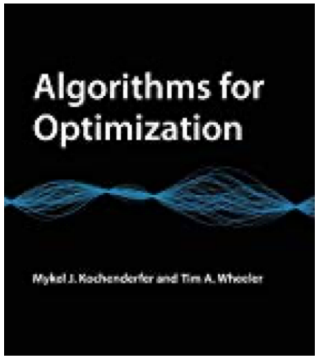
Books on Julia

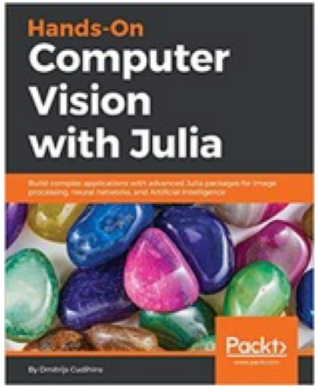




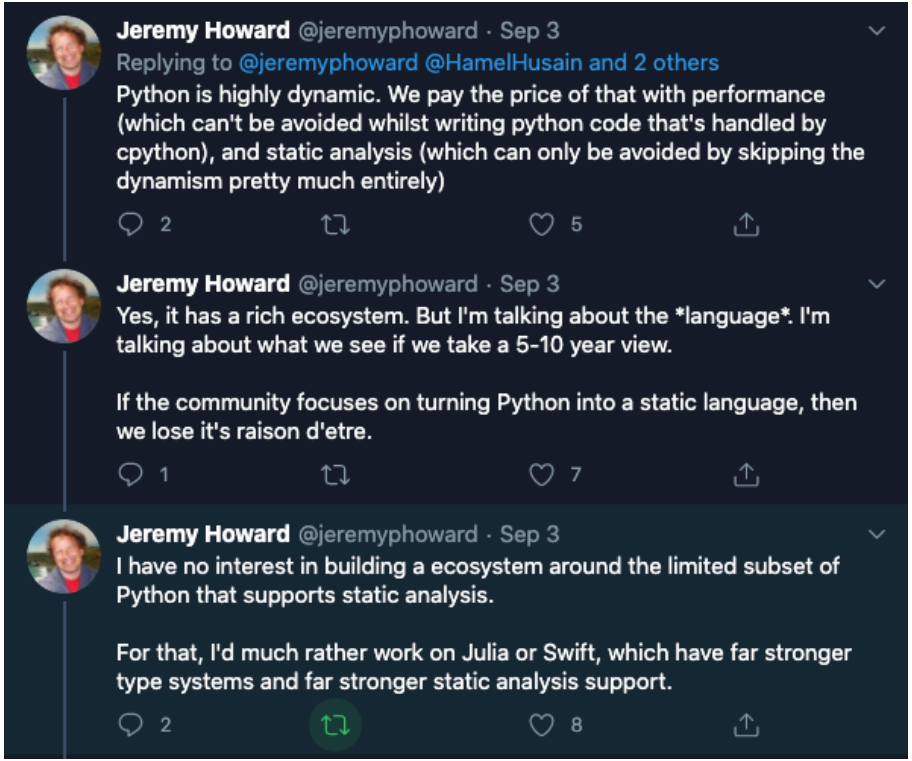




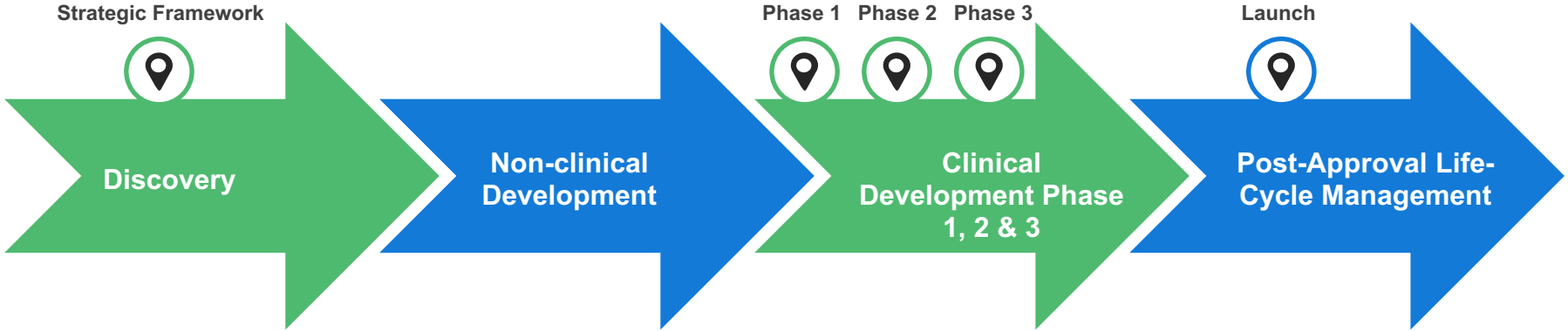




Google.ai Head Jeff Dean and Fast.ai co-author Jeremy Howard on Julia and Differentiable Programming



Varied analytics required along the drug development pipeline

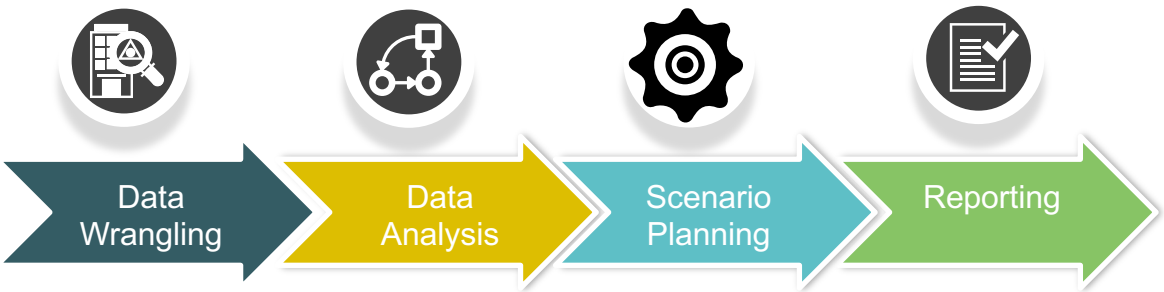


M&S Role

Discovery	Non-clinical Development	Clinical Development Phase 1, 2 & 3	Post-Approval Life-Cycle Management
Potency Projection QSP NLME	**Tox, Human Projections** NCA SP NLME PBPK	**Product Optimization, Dose, Efficay, Safety, Regulatory** NLME NCA BE IVIVC SP SML	**Dose, Efficay, Safety, Regulatory** NLME NCA BE IVIVC SP SML

NCA	Non-Compartmental Analysis	**PBPK**	Physiological Pharmacokinetics Modeling
NLME	Non-Linear Mixed Effects Modelin	**QSP**	Quantitative Systems Pharmacology Modeling
BE	Bioequivalence Testing	**SP**	Scenario Planning
IVIVC	In vitro In Vivo Correlation	**SML**	Scientific Machine Learning

High costs for interoperability



SAS
R
EXCEL

FORTRAN
C/C++
R
Matlab
Perl
Python

C/C++
R

R
Perl
Word

Efficient Modeling and Simulation → Accelerated Drug Development



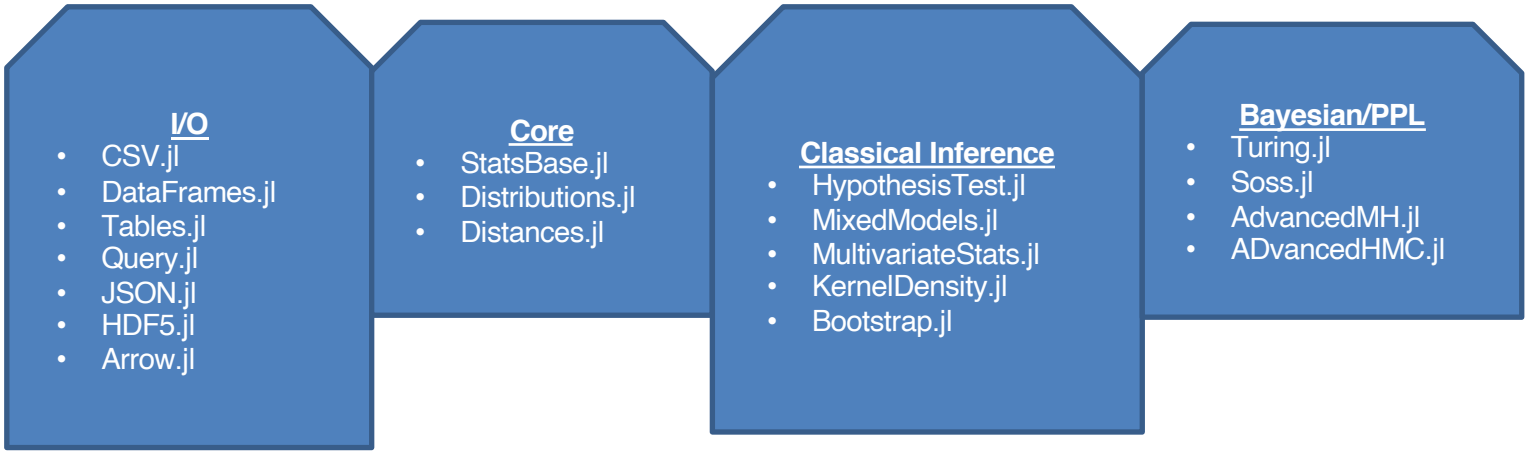

Pumas

100 % Julia

Julia has excellent interoperability with other languages, especially R

pumas

Statistics in Julia



Back to the Future: Lisp as a Base for a Statistical Computing System by Ihaka and Lang (2008)

Julia is actually a Lisp in disguise

Regulatory submissions with Pumas and Julia

Recently we worked with Dr Gobburu and his team on applying Model-informed Drug Development (MIDD) approach to support the development of one of our key assets. This work led to a successful meeting with the FDA and shortening the drug development program tremendously. All the modeling and simulation work submitted and reviewed by the Agency was performed using Pumas software. We are very pleased with the experience, quality of the work and the outcome.

SR. VICE PRESIDENT, PROPRIETARY PRODUCTS, DR. REDDY'S LABORATORIES LIMITED

Dr Anil Namboodiripad, PhD

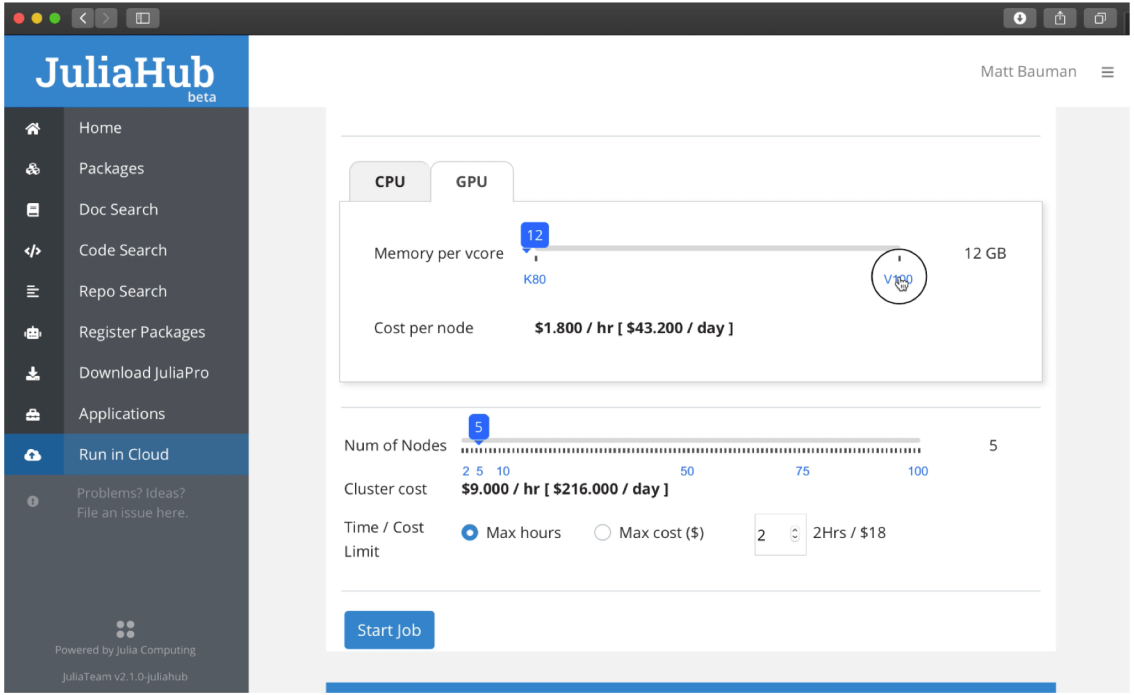
Key Highlights

Download Pumas at https://pumas.ai

- 450 budding pharmacists trained in Julia and Pumas in 2020 so far
- Over 40 pharma firms have taken up Julia and Pumas for their workflows
- Julia/Pumas has sped up multiscale models by 10-300x
- Julia/Pumas has allowed scientists to run on GPU
- 6 academic groups have shifted their educational programs and use Julia/Pumas along with R in their curriculum

JuliaHub with Pumas and much more

A Cloud Platform for Julia-based Pharmacometric workflows



- https://juliahub.com
- Single click parallelism integrated right into the IDE (Visual Studio Code)
- Complete browser-based workflow
- Support for polyglot multi-language workflows (R, Python, Fortran, Matlab, etc.)
- GPU acceleration
- CFR Part 11 compliance

Pfizer: 175x speedup in multi-scale QSP models

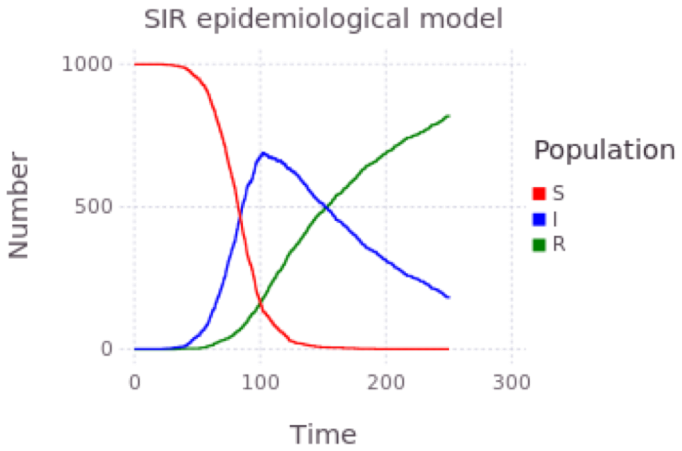
Best abstract award at ACOP 2020

- Improved time-to-market of a new drug by many months
- Because of limited shelf life of patent protection, this results in significant value
- Multi-million cost savings during drug development

Problem and Solution	Total Speedup Over Baseline
Cardiac MATLAB model, translated to Julia	26x
Parallelized Cardiac model runs	115x
Leucine model, Julia translation and ODE solver optimizations	7x per CPU, 175x per GPU 
Global optimization of Leucine model	13x
Virtual Population generation for Leucine model	**Cluster Scalable**
Beard model, Julia translation and ODE solver optimizations	2x per CPU
Global sensitivity analysis of the Beard model	**Multi-GPU Cluster Scalable** 

Gillespie algorithms in R and Julia

Implementation	Time per simulation (ms)
R (GillespieSSA)	894.25
R	1087.94
Rcpp	1.31
Julia (Gillespie.jl)	3.99
Julia (DifferentialEquations.jl)	1.20



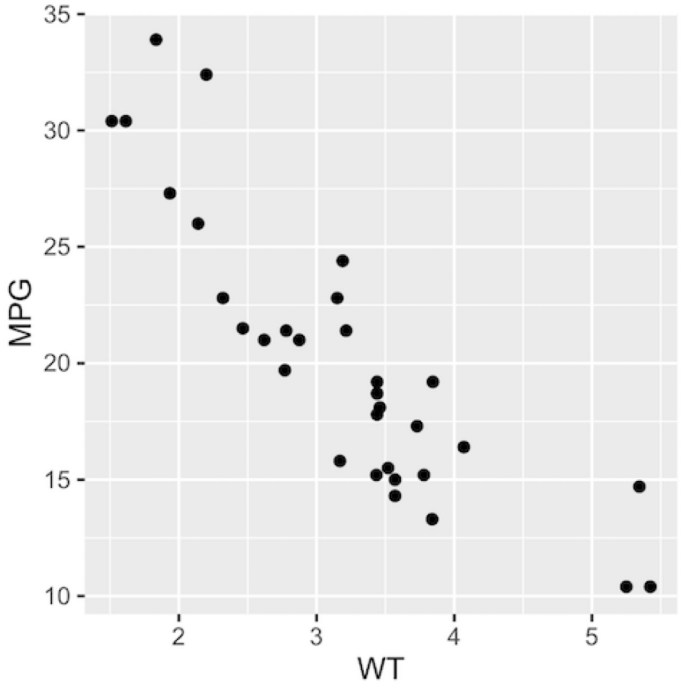
- Straightforward use of Julia's package is **225 times faster than R**
- Using DifferentialEquations.jl directly makes it **750 times faster than R**
- Historically getting speed like this required a second language (like C++)

Frost, Simon D.W. (2016) Gillespie.jl: Stochastic Simulation Algorithm in Julia. *Journal of Open Source Software* 1(3) doi:0.21105/joss.00042

Calling R from Julia

- Rcall.jl: https://github.com/JuliaInterop/RCall.jl
- Press $ to switch between Julia and R repl modes

```
julia> using RCall

julia> using RDatasets

julia> mtcars = dataset("datasets", "mtcars");

R> library(ggplot2)

R> ggplot($mtcars, aes(x = WT, y=MPG)) + geom_point()

 R>
```



Calling Julia from R

- The R JuliaCall library
  - https://cran.r-project.org/web/packages/JuliaCall/index.html

- Doug Bates
  - MixedModels.jl: https://rpubs.com/dmbates/377897
  - https://www.youtube.com/watch?v=oOd3JnEm3c8

- Chris Rackauckas
  - GPU-Accelerated ODE Solving in R with Julia, the Language of Libraries
  - https://www.stochasticlifestyle.com/gpu-accelerated-ode-solving-in-r-with-julia-the-language-of-libraries/

Pluto Notebooks

Writing a notebook is not just about writing the final document — Pluto empowers the experiments and discoveries that are essential to getting there.

Explore models and share results in a notebook that is:

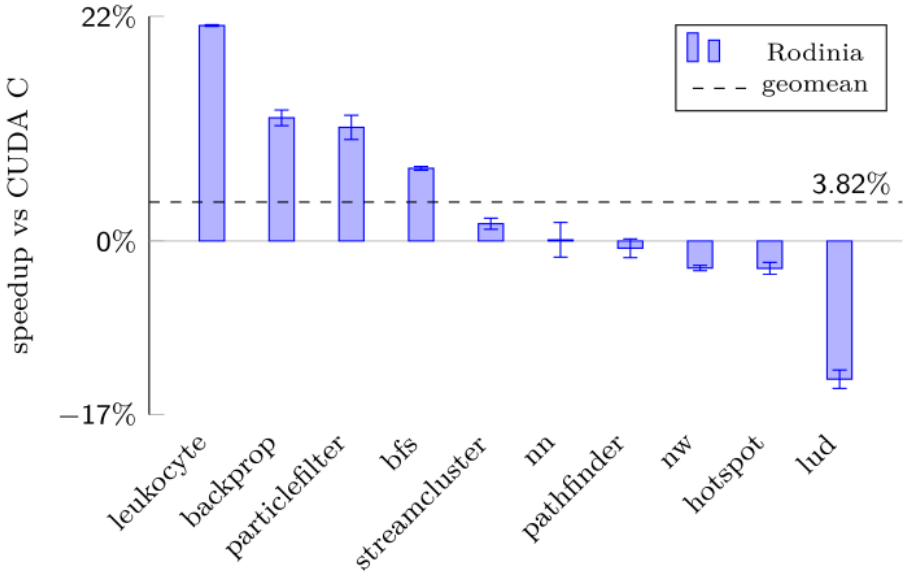
- ***Reactive*** - when changing a function or variable, Pluto automatically updates all affected cells.
- ***Lightweight*** - Pluto is written in pure Julia and is easy to install.
- ***Simple*** - no hidden workspace state; friendly UI.

JuliaCon 2020 talk:
https://www.youtube.com/watch?v=IAF8DjrQSSk

Pluto.jl

Julia on GPUs: https://juliagpu.org

Supports NVIDIA GPUs. Nascent support for AMD and Intel GPUs.



Benchmarks compared to CUDA C

Noteworthy new capabilities

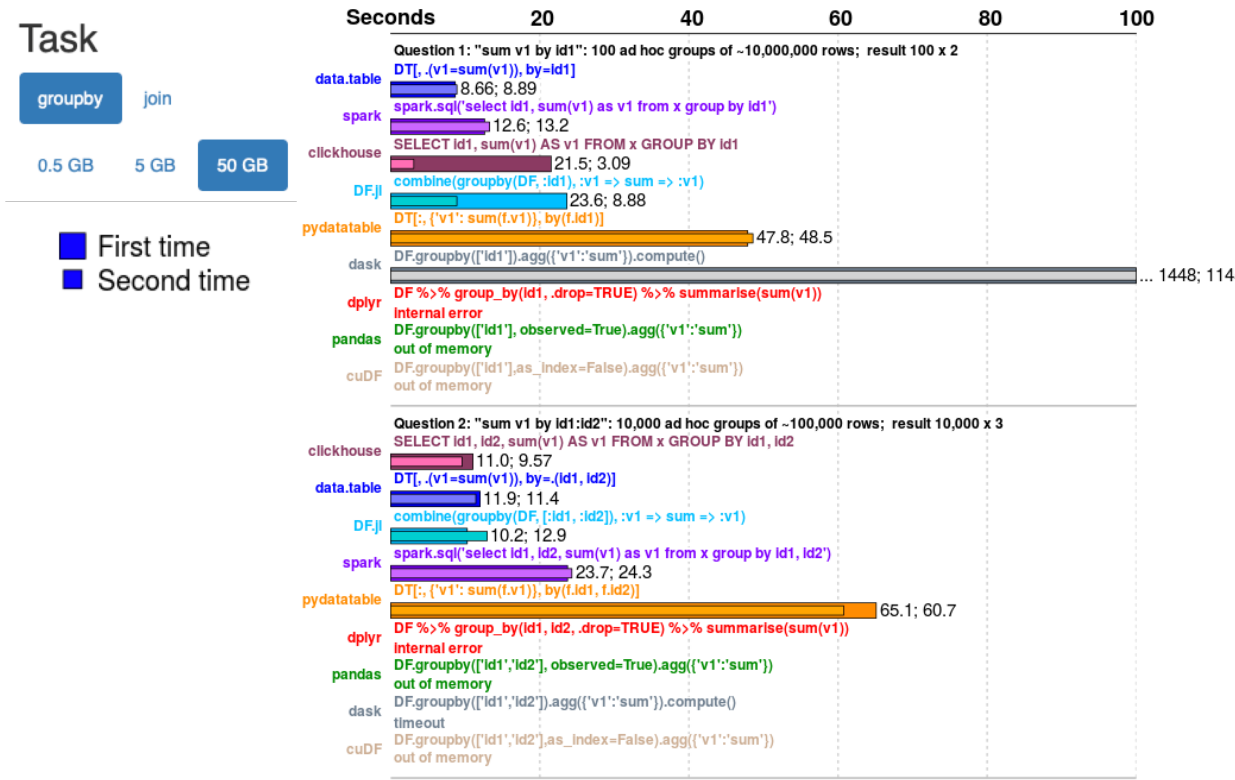
- Multi-GPU programming
- Support for CUDA 11 (and CUDA 10 also)
- CUDNN support
- Multi-tasking and multi-threading

Noteworthy applications

- 300x improvement in pharmaceutical workloads
- 1,000 GPU parallel deployment at CSCS (Switzerl
- Clima Project – Oceananigans.jl
- Multi-physics simulations
- Reinforcement learning – AlphaZero.jl

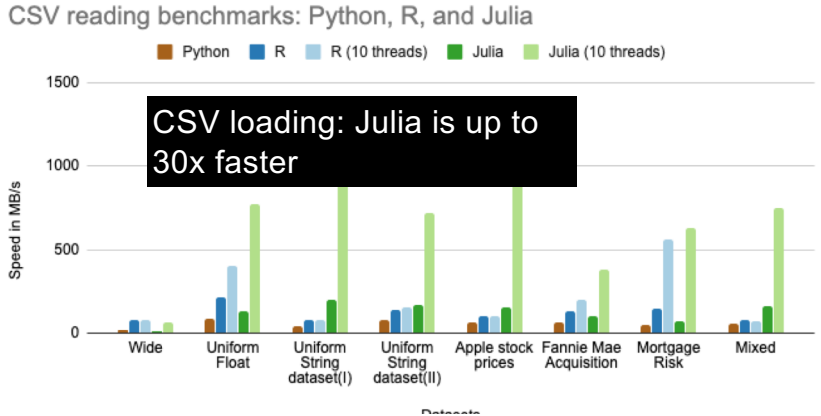
Julia processes your data quickly (DataFrames.jl)

- Julia can handle very large datasets
- Processing 50 GB of data on a machine with 128GB of RAM is fast!



https://h2oai.github.io/db-benchmark/

Data Science with DataFrames.jl and CSV.jl



Input table: 100,000,000 rows x 9 columns (5 GB)

ClickHouse	20.3.8.53	2020-04-30	7s
data.table	1.12.9	2020-04-22	13s
spark	2.4.5	2020-03-04	36s
DataFrames.jl	0.21.1	2020-05-24	42s
(py)datatable	0.11.0a0	2020-05-24	68s
pandas	1.0.3	2020-05-19	92s
dplyr	0.8.5	2020-03-08	191s
d			305s
c			of memory
M			pending

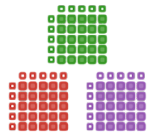
H2O DataFrames benchmark: GroupBy
Julia is 4x faster than dplyr.

DataFrames.jl

- High performance native Julia package for data manipulation
- GroupBy operations 2x faster than Pandas and R
- DataFrames.jl status and roadmap

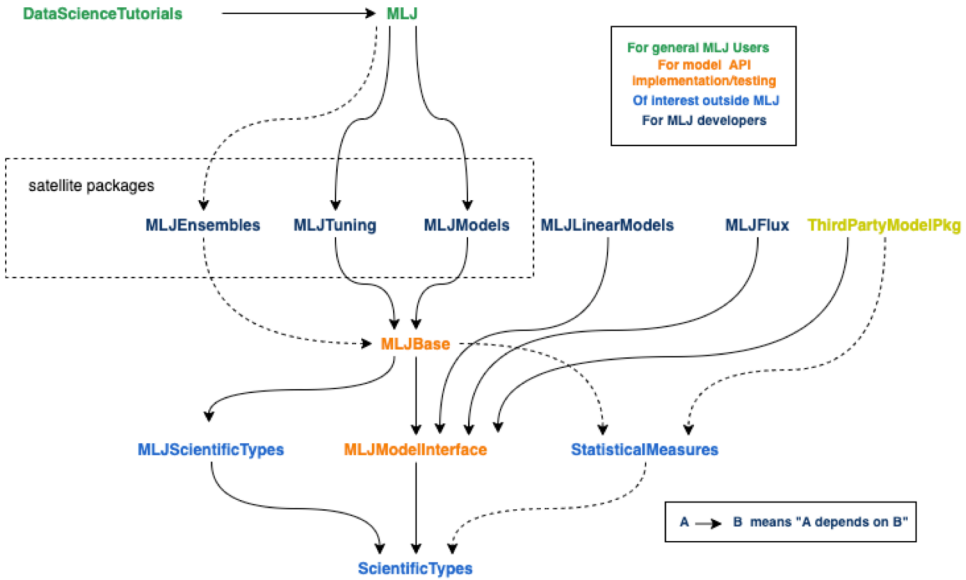
CSV.jl

- Native Julia package for loading CSV files
- Significantly faster than Python (pandas) and R
- Multi-threaded
- The great CSV showdown (TowardsDataScience)



Machine Learning with MLJ.jl

The scikit-learn like package in Julia

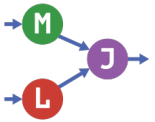


Dependency chart for MLJ repositories. Repositories with dashed connections do not currently exist but are planned/proposed.

[MLJ.jl: https://github.com/alan-turing-institute/MLJ.jl](https://github.com/alan-turing-institute/MLJ.jl)

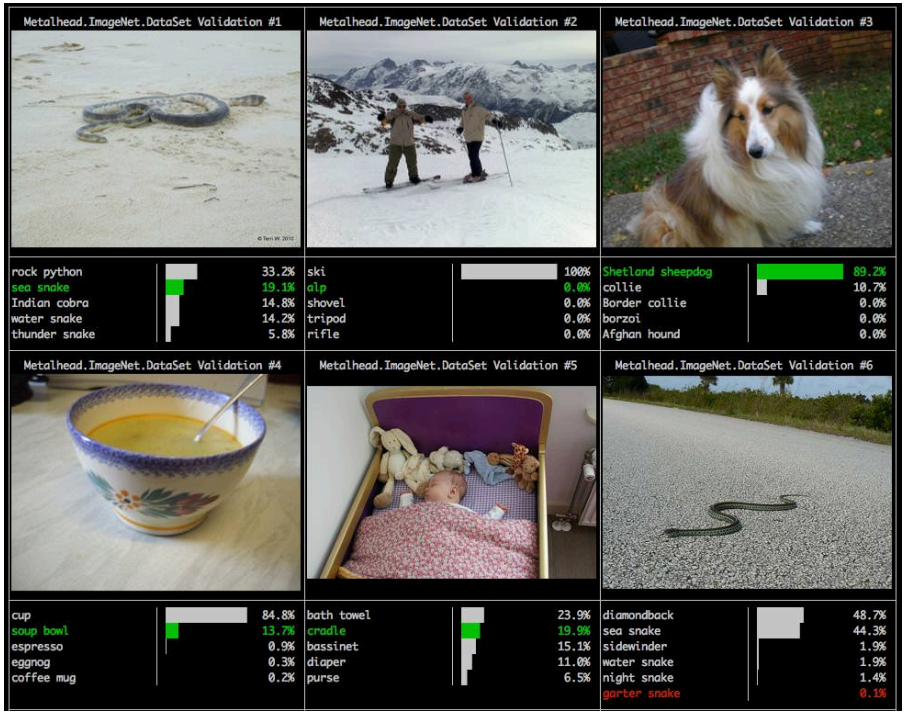
Packages supported by MLJ.jl

Package	Maturity
Clustering.jl	high
DecisionTree.jl	high
EvoTrees.jl	medium
GLM.jl	medium
LIBSVM.jl	high
LightGBM.jl	high
MLJFlux.jl	experimental
MLJLinearModels.jl	experimental
MLJModels.jl (builtins)	medium
MultivariateStats.jl	high
NaiveBayes.jl	experimental
NearestNeighbors.jl	high
ParallelKMeans.jl	experimental
ScikitLearn.jl	high
XGBoost.jl	high



Deep Learning with Flux.jl

https://fluxml.ai




```
using Flux

W = rand(2, 5)
b = rand(2)

predict(x) = (W * x) .+ b
loss(x, y) = sum((predict(x) .- y).^2)

x, y = rand(5), rand(2) # Dummy data
l = loss(x, y) # ~ 3

θ = params([W, b])
grads = gradient(() -> loss(x, y), θ)
```

Pre-trained models

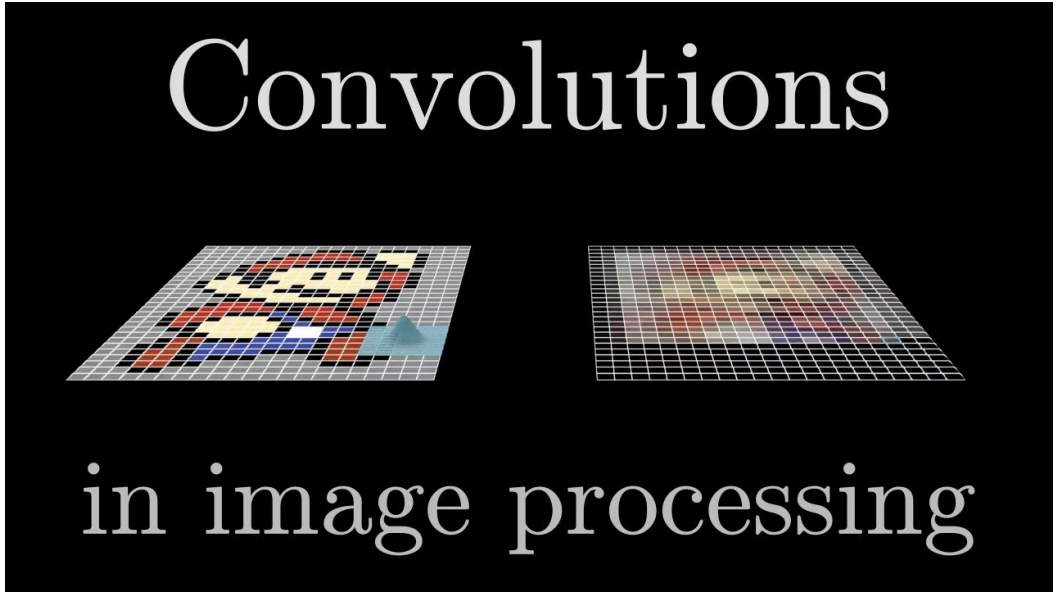
- MetalHead.jl
- YOLO.jl
- Darknet.jl
- ObjectDetector.jl
- Transformers.jl
- TextAnalysis.jl
- GeometricFlux.jl

Flux.jl

- Simple Kera—like syntax
- Native Julia implementation
- Easy to look under the hood and modify
- Model Zoo to get started

Image Processing with Images.jl

https://juliaimages.org



https://www.youtube.com/watch?v=DGojI9xcCfg
https://www.youtube.com/watch?v=8rrHTtUzyZA

Images.jl

JuliaImages is focussed on a clean architecture and hopes to unify "machine vision" and "biomedical 3d image processing" communities.

- Native Julia based image processing package
- Being used at MIT in a class taught in
  collaboration with 3blue1brown
- Native Julia datatypes such as RGB.
- Extensive documentation

Optimization

INFORMS Computing Society Prize

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

JuMP.jl, Optim.jl

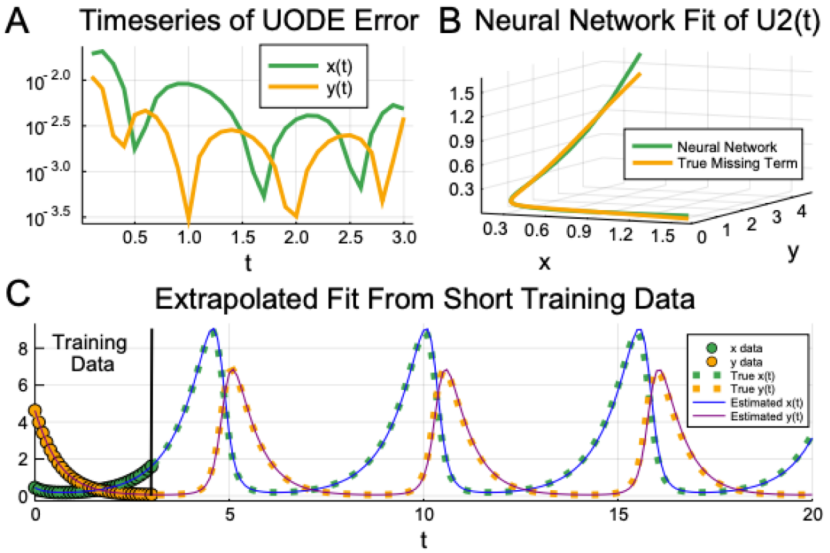
- JuMP is a DSL for mathematical optimization
- Received INFORMS Computing Society Prize
- Supports over 25 backend solvers
- Optim.jl provides univariate and multivariate optimization in Julia.
- Jump-Dev 2020 is in progress

JuMP.jl: https://github.com/IainNZ/SudokuService

JUMP

Scientific Machine Learning

https://sciml.ai



Benchmarks compared to CUDA C

Noteworthy new capabilities

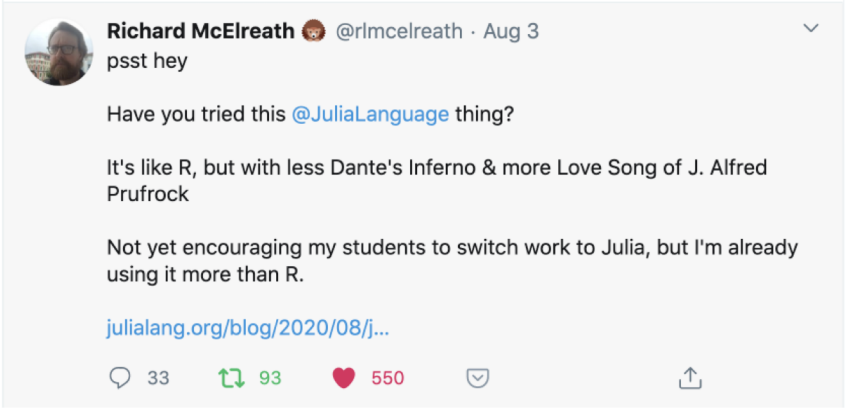
- Combine Science and Machine Learning
- Comprehensive Differential Equation Solvers
- GPU acceleration
- MTK.jl: A DSL for modeling and simulation
- Automatic-differentation

Noteworthy applications

- ACED: Computational Electrochemical Systems
- Clima: Climate Modelling
- NYFed DSGE modelling
- Pumas.jl: Pharma simulations
- MADS: Decision Support System
- QuantumOptics
- Robotics

Probabilistic Programming

https://turing.ml




```
@model gdemo(x, y) = begin
  # Assumptions
  σ ~ InverseGamma(2,3)
  μ ~ Normal(0,sqrt(σ))
  # Observations
  x ~ Normal(μ, sqrt(σ))
  y ~ Normal(μ, sqrt(σ))
end
```

Turing.jl

Intuitive
Turing models are easy to read and write — models work the way you write them.

General-purpose
Turing supports models with discrete parameters and stochastic control flow. Specify complex models quickly and easily.

Modular
Turing is modular, written fully in Julia, and can be modified to suit your needs.

High-performance
Turing is fast.

Thank you

Contact us: info@juliacomputing.com

Julia SURE

Enterprise support and indemnity

Julia TEAM

Enterprise governance and collaboration

Julia RUN

Deployment and scalability

Discover The Power of Julia

Our products make Julia easy to use, easy to deploy and easy to scale

Julia PRO

Seamless development

Pumas

Pharmaceutical modelling and simulation