Key features Overview

Syntax

Package

Language binding

Speed

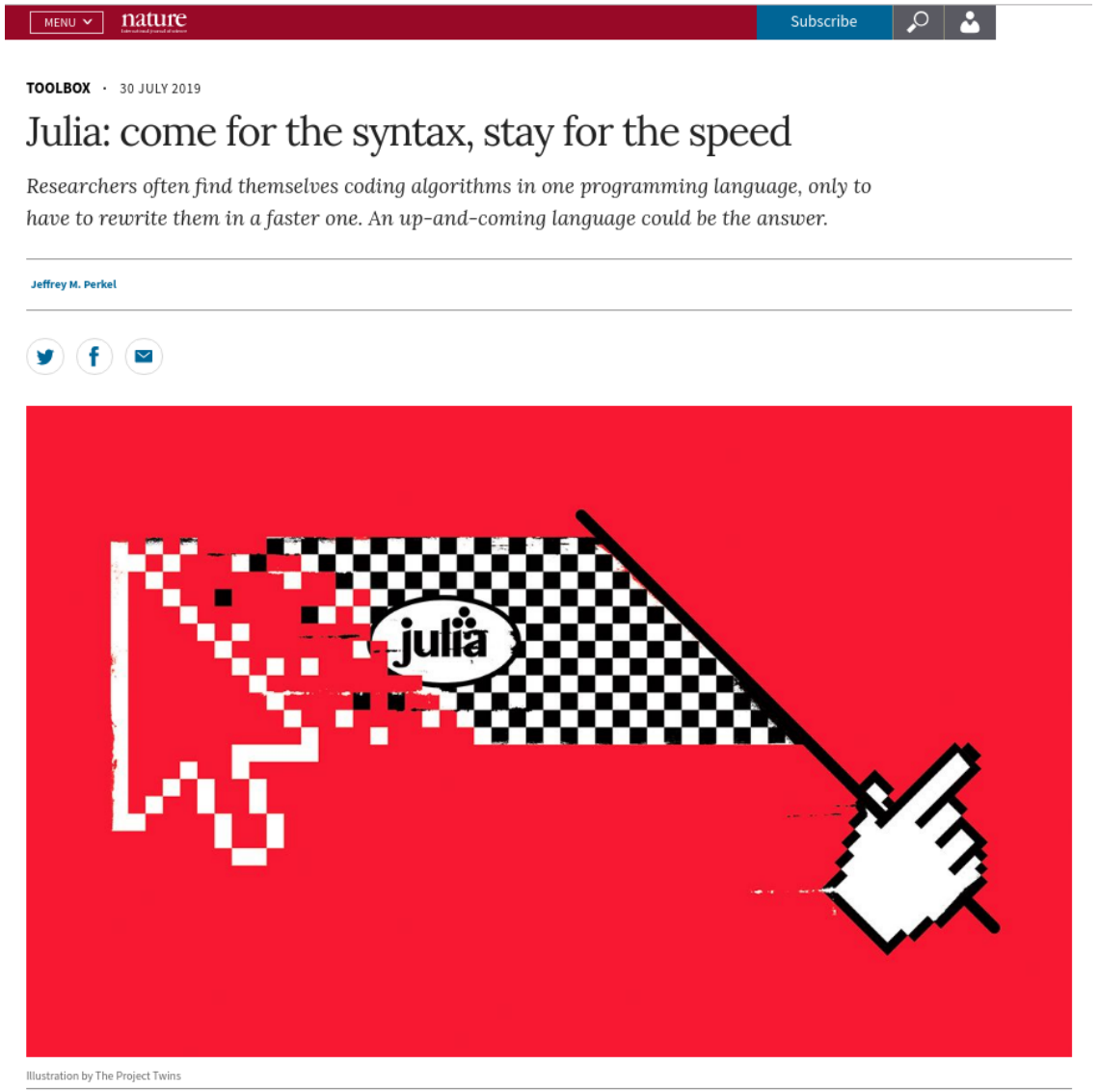
Why Julia is fast and flexible

Parallelization(+ GPU)

Julia Syntax

Recent article (30 July 2019) in Nature journal about 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.*"(Jeffrey M. Perkel)



Jeffrey M. Perkel, "**Julia: come for the syntax, stay for the speed**", Nature. 2019 Aug;572(7767):141-142. doi: 10.1038/d41586-019-02310-3.

```
function MySum(x::Vector{Float64})
    total = 0.0
    for i in 1:length(x)
        total += x[i]
    end
    total
end
```

```
MySum (generic function with 1 method)
```

```
a = randn(100)
MySum(a)
```

```
10.204973501090725
```

```julia
# Notes: No extra Package to be installed to write with symbol character.
function ∑(X)
    total = zero(eltype(X))
    for xᵢ in X
        total += xᵢ
    end
    total
end
```

```
∑ (generic function with 1 method)
```

```julia
a = randn(100)
∑(a)
```

```
-16.474855643307926
```

Matlab, Python and Julia Comparison

Creating Vectors

MATLAB	PYTHON	JULIA
Row vector: size (1, n)		
`A = [1 2 3]`	`A = np.array([1, 2, 3]).reshape(1, 3)`	`A = [1 2 3]`
Column vector: size (n, 1)		
`A = [1; 2; 3]`	`A = np.array([1, 2, 3]).reshape(3, 1)`	`A = [1 2 3]'`
1d array: size (n,)		

Not possible	`A = np.array([1, 2, 3])`	`A = [1; 2; 3]`
		or
		`A = [1, 2, 3]`

Integers from j to n with step size k		
`A = j:k:n`	`A = np.arange(j, n+1, k)`	`A = j:k:n`
Linearly spaced vector of k points		
`A = linspace(1, 5, k)`	`A = np.linspace(1, 5, k)`	`A = range(1, 5, length = k)`

Matlab, Python and Julia Comparison

Creating Matrices

MATLAB	PYTHON	JULIA
Create a matrix		
`A = [1 2; 3 4]`	`A = np.array([[1, 2], [3, 4]])`	`A = [1 2; 3 4]`
2 x 2 matrix of zeros		
`A = zeros(2, 2)`	`A = np.zeros((2, 2))`	`A = zeros(2, 2)`
2 x 2 matrix of ones		
`A = ones(2, 2)`	`A = np.ones((2, 2))`	`A = ones(2, 2)`
2 x 2 identity matrix		

`A = eye(2, 2)`	`A = np.eye(2)`	`A = I # will adopt`
		`# 2x2 dims if demanded by`
		`# neighboring matrices`

Diagonal matrix		
`A = diag([1 2 3])`	`A = np.diag([1, 2, 3])`	`A = Diagonal([1, 2,` `3])`
Uniform random numbers		
`A = rand(2, 2)`	`A = np.random.rand(2, 2)`	`A = rand(2, 2)`

Julia Syntax

Programming language comparison resources:

- Noteworthy Differences from other Languages
- Julia Vs R Comparison Cheat Sheet
- MATLAB–Python–Julia cheatsheet
- The Matrix Cheatsheet Matlab–Python-R–Julia comparison

Julia Packages

Julia has over 2000 registered packages. They are a substantial part of the Julia ecosystem.

To explore existing packages related to your area of interest you can check out the following link: https://pkg.julialang.org/docs/

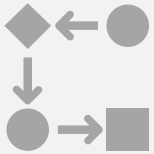
Julia comes with a built-in package manager named Pkg, but it also com with an interactive Package Mode: Example

Julia Packages Worth Mentioning

- **Statistics and Math:** `StatsBase` (Basic functionalities for statistics), `LinearAlgebra` (Linear algebra operations), `Distributions` (Probability distributions), `GLM` (Generalized linear models)
- **Data tools:** `DataFrames` (Essential tools for tabular data), `Queryverse` (A meta package for data science), `CSV` (For working with CSV files), `XLSX` (Excel file reader and writer)
- **Biology:** `Bio` (Framework for computational biology and bioinformatics)
- **Machine Learning:** `Flux` (ML library), `TensorFlow` (A wrapper around TensorFlow), `Knet` (Deep learning framework), `FScikitLearn` (Scikit-learn framework)
- **Plot:** `Plots` (Julia visualizations and data analysis), `PyPlot` (matplotlib plotting like), `Gadfly` (ggplot2 plotting like), `UnicodePlots` (Unicode-based scientific plotting for working in the terminal)
- **Language binding:** `PyCall` (For Python calling), `RCall` (For R calling)
- **Jupyter:** `IJulia` (Julia kernel)

Language Binding

Even though, the package ecosystem still has room to grow. Julia has excellent foreign function interfaces. Easily call into other languages such as python with with `PyCall` or R with `Rcall`.



This means that you don’t have to wait until the Julia ecosystem is fully mature, and that you don’t have to give up your favorite package/library from another language when moving to Julia!

Julia Speed

"*The benchmarks are written to test the performance of identical algorithms and code patterns implemented in each language.*"(https://julialang.org/benchmarks)