Why Julia?

Johanni Brea

Introduction à l'apprentissage automatique

GYMINF 2021

A Statement of the Founders of 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 are power Matlab users. Some of us are Lisp hackers. Some are Pythonistas, others Rubyists, still others Perl hackers. There are those of us who used Mathematica before we could grow facial hair. There are those who still can't grow facial hair. We've generated more R plots than any sane person should. C is our desert island programming language.

We love all of these languages; they are wonderful and powerful. For the work we do — scientific computing, machine learning, data mining, large-scale linear algebra, distributed and parallel computing — each one is perfect for some aspects of the work and terrible for others. Each one is a trade-off.

We are greedy: we want more.

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?)

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

A User's Perception


```
library(tRucks)
```

```
#include <stdlib.h>
#include <jigsaw.h>
#include <lathe.h>
```

```
using Trucks
```

```
using Wheels
using Windows
using Technic
```

https://discourse.julialang.org/t/what-is-the-advantage-of-julia-over-fortran/65964/101

Why I Use Julia

None of the previous languages I used (C, C++, C#, Fortran, Python, Matlab, Mathematica) had the same convincing combination of features like Julia:

- Easy, rapid, interactive prototyping.
- Smooth transition to high-performance libraries.
- High flexibility, modularity and compositionality.
- Great tools for testing and reproducibility.
- Open Source.

Annoying Features of Julia

- Almost no code is fully precompiled, but there is always some compilation happening when calling a function for the first time. E.g. from launching a julia script until seeing the first plot, it may take several seconds (”Time to first plot”).
- There is no easy way to create small executables or compiled libraries.
- The ecosystem is still young: many packages are ”rough around the corners”.
- The community is still young: not so many questions/answers on the internet. However, I perceive the community a is very helpful `julialang.org/community`.

Links

- Pluto and how it is used at MIT
  `https://youtu.be/IAF8DjrQSSk`
  `https://computationalthinking.mit.edu`
- The unreasonable effectiveness of multiple dispatch
  `https://youtu.be/kc9HwsxE1OY`
- Dispatching Design Patterns
  `https://youtu.be/n-E-1-A_rZM`
  `https://github.com/ninjaaron/dispatching-design-patterns`