The article "The unreasonable effectiveness of the Julia programming language" was written by Lee Philips for the site "Arstechnica"

<u>The summary is :</u>

First : The problem
Then : What is possible to do with Julia
Finnaly : How Julia works

One of the main problems that Julia answers is the problem of the two languages.
One to quickly program a prototype (like Python), the other seeking to optimize performance (like C).

This new language was designed to be able to do machine learning, data mining, distributed and parallel computing.

To be able to do all this, we had to combine the advantages of C, Ruby, Lisp, Matlab, Python, R and Perl.

So what's possible and what's not to do with Julia ?

Julia is not made to do web sites and statistics (because it’s far behind R)

Julia is however very powerful for mathematical optimization, image manipulation, machine learning, simulation of natural phenomenon

A good practical example is modeling the reacting flow inside a rocket engine combustor.

Julia works with a just in time compilation : each time the user enters code into the interpreter, the interpreter is compiled and runs as machine code.

The heart of the language is written in C, the parser in Scheme and the majority of the standard library in Julia.

Julia's typing is strong, dynamic and inferred
* **strong** : ensures that the data types used correctly describe the data being manipulated.
* **dynamic** : typing "on the fly", during the execution of the code.
* **inferred** : automatically search for types associated with expressions, without them being explicitly indicated in the source code.

Julia is an object oriented and functional programming language. All objects are first class citizens (which can be used without restriction). There are several types :
- abstract type (abstract)
- mutable struct (normal)

- **struct** (unchangeable)

The types can be configured.

Another of the concepts that makes the Julia language so innovative is Multiple dispatch : a function can be specialized for more than one of its formal parameters. As we do in software engineering with mister DONY.

It is never necessary to specify the types and the most specialized function will always be called during execution.

The functions are compiled on the fly for each set of arguments, which allows each time to have optimized native code (assembler), viewable with the “code_native” function.

There is also the possibility to use functions written in C with Julia.

A concept that can greatly optimize performance is parallel programming in distributed memory.
The objective is to realize the greatest number of operations in the shortest possible time.

There are also packages for calling Python, Java and R code.

---

My personal opinion is Julia is a very promising language that can be as simple as a high level language while being as powerful as a low level language. To perform simulations and process data for scientific purposes.

<u>Vocabulary test :</u>

- **effectiveness** = efficacité
- **on the fly** = à la volée
- **set of** = ensemble de
- **viewable** = visible
- **promising** = prometteur
- **to process** = traiter
- **purposes** = le but
- **scalable** = évolutif
- **computational scientist** = scientifique informatique
- **uncertainty quantification** = quantification de l'incertitude