Evaluating the Julia Programming Language

•••

Sean Seruya, Frank Mai, Robert Wolfe, William Zabet

Criteria for Evaluating

- *Rapid Development*
- *Efficiency/Performance*
- *Learnability*
- *Functionality*
- *Reliability and Safety*
- *Simplicity*
- *Abstraction*

Rapid Development

Programmers are more expensive than machines, so they'd better be able to make fast progress. (We should consider both the language and its environment in making this evaluation.)

Aspects for Rapid Development

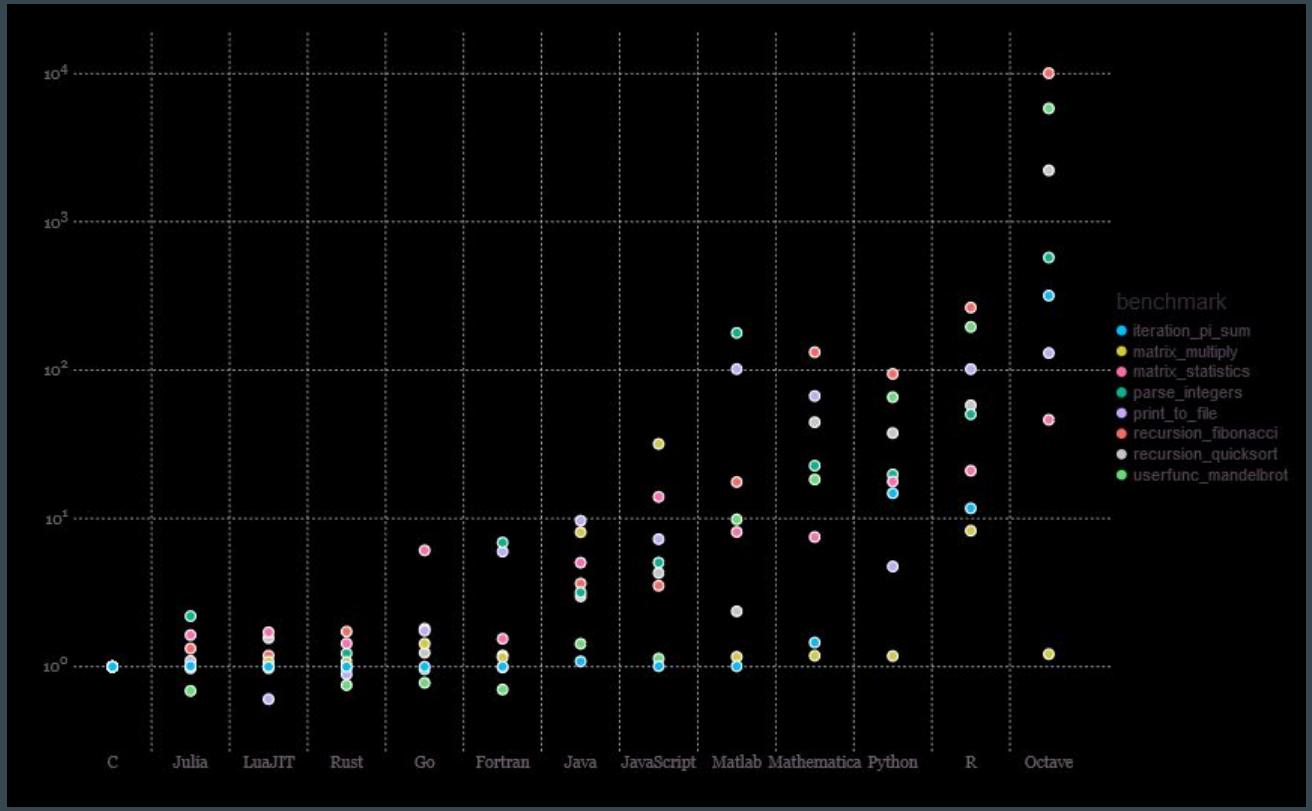
	Julia	Python	C	C++	Java
Number of Google search results	~118 million	~457 million	~6.4 billion	~ 86 million	~352 million
Number of YouTube videos	~64.7 million	~59 million	~3.4 billion	~641,000	~84 million
Stackoverflow posts/questions	18,579*	1,465,382	333,626	678,312	1,687,326
Github Repos	23,596	1,499,423	9,418,562		1,475,468

Efficiency/Performance

The compiler should be fast. The code itself should be fast.

https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

Mini Benchmarks



Performance

	Julia	Python	C	C++	Java
Regex-redux	3.04	2.67	1.42	1.86	10.27
Mandelbrot	2.64	263.87	1.64	1.51	6.84
n-body	9.57	891.12	7.30	7.70	21.85
spectral-norm	2.79	170.10	1.98	1.99	4.15
pidigits	2.06	2.39	1.75	1.82	1.83
k-nucleotide	7.55	72.58	4.84	3.88	9.14

Learnability

Learning Resources:

Tutorial Resources	C*	C++	Java	Python	Julia
Youtube Videos	8.75 million	444,000	3.59 million	2.82 million	132,000
Udemy	260	283	934	1,617	130
Amazon	4000+	2000+	3000+	3000+	156
Google	361 million	83.3 million	162 million	118 million	1.75 million

*the results for c could possibly contain results for c++ as well

Functionality

Julia Language is best for scientific and numerical computing

- Data Science
- Machine learning
- Visualization

Functionality

topuniversities.com

1. A.I. and robotics
2. Big Data Analytics
3. Computer-assisted education
4. Bioinformatics
5. Cyber security

Functionality

phdassistance.com

1. Data Warehousing
2. IoT
3. Big Data
4. Cloud Computing
5. Semantic Web
6. MANET
7. Machine Learning
8. A.I.
9. Data Mining...

Functionality

twinword.com/ideas

- Big data - 135k average monthly searches
- Data Analytics - 74k
- Data Science - 74k
- Machine learning - 60.5k

Functionality

ubuntupit.com

1. Big Data Engineer
2. Data Scientist
3. Information Systems Security Manager
4. Data Architect
5. Data Security Analyst
6. Application Architect
7. Data Manager

Functionality

glassdoor.com

1. Data Scientist - $117k
2. Machine Learning Engineer - $114k
3. Systems Architect - $105k
4. IT Project Manager - $97k
5. Cloud Engineer - $95k
6. Mobile App Developer - $92k
7. UX Designer - $90k
8. Software Developer - $80k
9. Database Administrator - $80k
10. Cybersecurity Analyst - $79k; 11. Web Developer - $75k; 12. Game Designer - $73k

Reliability and Safety

Reliability and safety

When computers go down, planes crash, phone systems break, nuclear reactors melt down, cash machines close. We'd like to avoid this.

- Used across many companies for projects where reliability is crucial
- Unlimited Scalability
- Solves the two language problem

Simplicity

Simplicity

> The language should have a minimal number of primitive concepts/features.

- Multiple Dispatch, Metaprogramming, Unicode Symbols, etc.
- Flexible Syntax
- Ease of Integration
- Elegant Code / Less lines of Code

Abstraction

Abstract data types are syntactic enclosures that include only the data representation of one specific data type and subprograms for operating on that type

Measures of a language with robust support for Abstraction include:

- Support for abstract data types (classes)
- Encapsulation
- Inheritance
- Polymorphism

Abstraction: Language Comparison

	Julia	Python	C	C++	Java
Classes / ADTs	No classes, but has ADTs; ADT methods defined outside type definition	Yes	No	Yes	Yes
Encapsulation	Methods not encapsulated	Yes	No	Yes	Yes
Inheritance	Supports hierarchy of ADTs	Yes	No	Yes	Yes
Polymorphism	Parametric polymorphism using multiple dispatch	Yes	No	Yes	Yes

Julia Abstraction Summary

Julia includes abstract data types, which can be structured hierarchically, but does not have traditional object-oriented classes

Julia methods must be defined outside of types to allow for multiple dispatch; thus, they cannot be encapsulated with a type

Multiple dispatch provides polymorphic method functionality

Questions?