THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

CSCI-6221
Julia Group Presentation

Alan(Changjia) Yang
Yvonne(Youwen) Liu
Robert(Bo) Liu
Waad Algorashy
Tiffany Nguyen

Evaluation Criteria

- Community
- Package Ecosystem
- Learning Materials
- Employability
- Programming Paradigms
- Performance
- Concurrency

Community

The Julia community has a presence across multiple platforms such as GitHub Twitter, LinkedIn,Facebook ..etc.

Community

Measured by:

- # projects actively developed in total on GitHub
- # Conferences in history
- # Hacker News Posts in total
- “Programming, scripting, and markup languages” question of Stack Overflow survey
- “Most Loved” question of Stack Overflow survey
- “Top Paying Technology” question of Stack Overflow survey

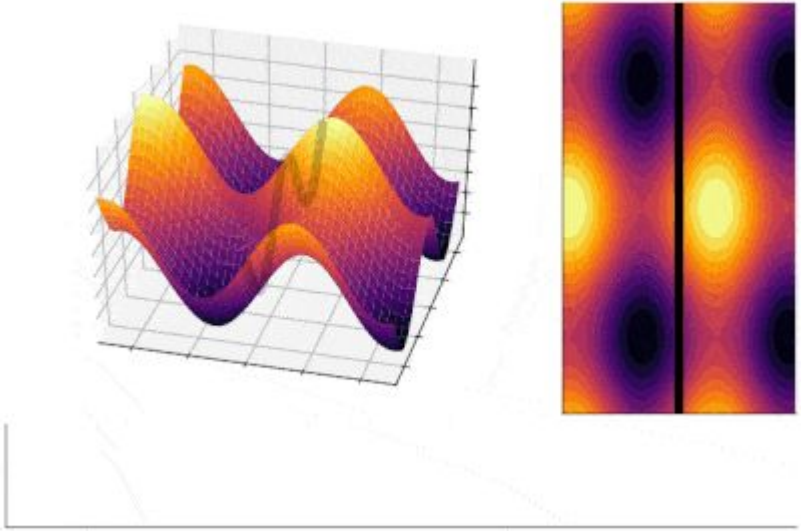
Note: The results of “Most Loved” question , “Programming, scripting, and markup languages” question and “Top Paying Technology” question of Stack Overflow survey is based on 2021 survey results. The result of “Top Paying Technology” is median number of salary.

Community

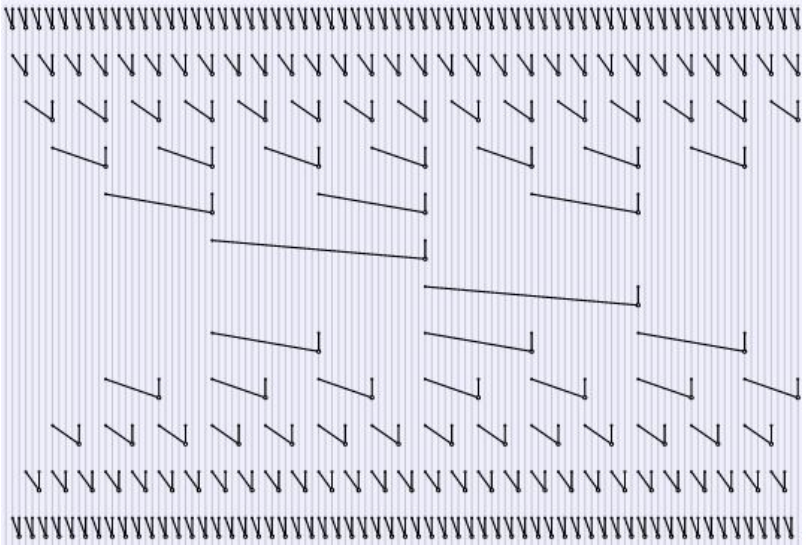
	Java	JavaScript	C++	Python	Julia
GitHub Projects	2,295,514	1,254,871	878,819	**2,499,338**	38,149
Conferences	**2097**	22	4	22	5
Hacker News Search	17,162	**152,358**	11,368	161,253	6,262
Commonly used languages	35.35%	**64.96%**	24.31%	48.24%	1.29%
StackOverflow "Most Loved" Rating	47.15%	**72.73%**	49.24%	67.83%	70.69%
StackOverflow "Top Paying Technology"	$51,888	$54,049	$54,049	$59,454	**$65,228**

Package Ecosystem

Visualization:
Data Visualization and Plotting

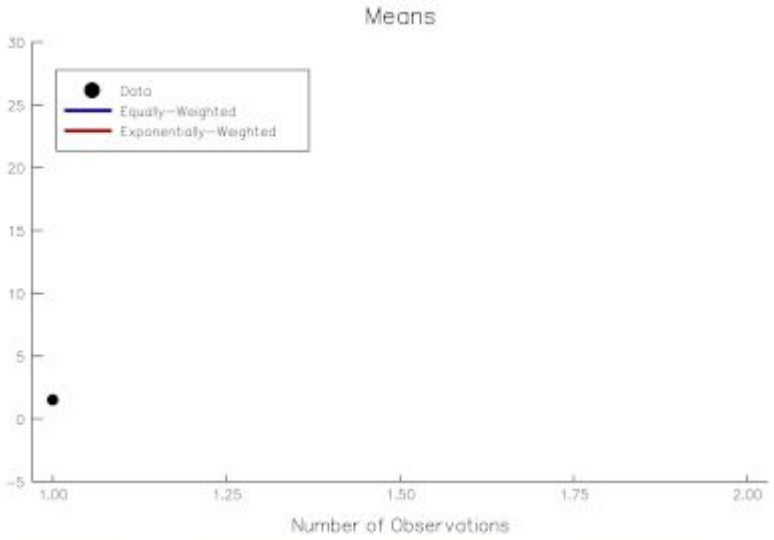


Parallel Computing:
Parallel and Heterogeneous Computing

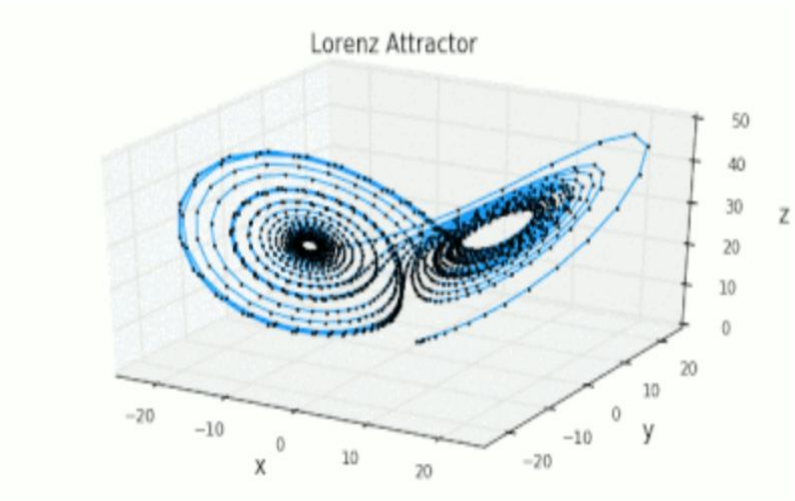


Package Ecosystem

Data science:
Interact with your Data



Scientific domain:
Scientific Computing



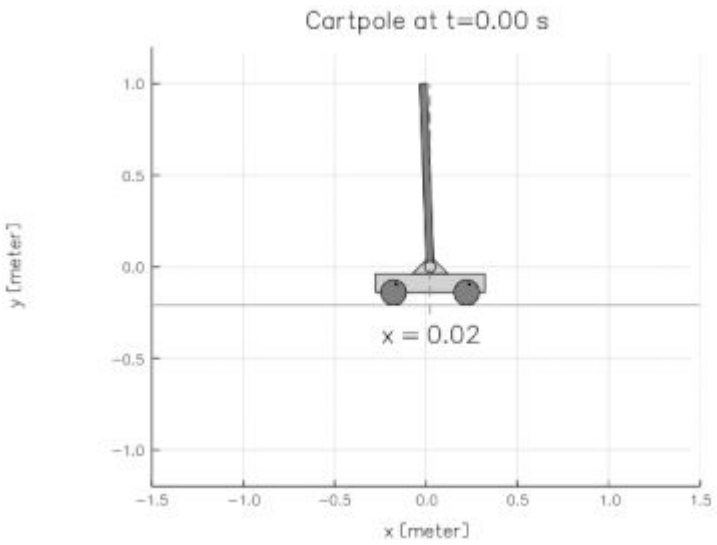
Package Ecosystem

Machine learning :

Scalable Machine Learning ;

MLJ.jl package, which include generalized linear models, decision trees, and clustering.

Flux.jl and Knet.jl packages for Deep Learning.



Package Ecosystem

Measured by:

- Number of packages in the dominant package manager as scraped by modulecounts.com

Package Ecosystem

	Java	JavaScript	C++	Python	Julia
Modules Count	467,604	**1,927,358**	329	368,258	4,385

Notes:

- Maven is considered the de-facto Java package manager.
- The package manager of python is PyPi.
- C/C++ does not have a centralized package ecosystem. This is using Clibs, which was created by a longtime JavaScript dev.

Learning Materials

- **Official documents**

  Ex: julialang.org

- **Online video tutorials**

  Ex: Youtube

- **Online courses:**

  Ex: Udemy

- **Online projects:**

  Ex: Github, Stackflow

- **Books.**

Learning Materials

Measured by

- Number of Books returned by search term “xxx programming language” on Amazon
- Number of courses on Udemy

Note: Amazon book store does not show the specific number over 1000 results

Learning Materials

	Java	JavaScript	C++	Python	Julia
Amazon Books	**10000+**	8000+	9000+	**10000+**	498
Udemy Courses	970	371	355	**1,499**	19

Note: Amazon book store does not show the specific number over 1000 results

Employability

- **Data Analyst & Data scientist :**

  Advanced understanding of a statistical programming language such as R, Python, or Julia.

- **Systems Programmer**

  Julia Computing.

Employability

Measured by number of impressions of “xxx developer” on popular job sites located in United States:

- Indeed
- ZipRecruiter
- LinkedIn

Employability

	Java	JavaScript	C++	Python	Julia
Indeed	**103,526**	90,924	25,990	82,648	110
ZipRecruiter	201,640	145,415	**340,675**	296,068	2,423
LinkedIn	870,000	809,000	782,000	**902,000**	1,000

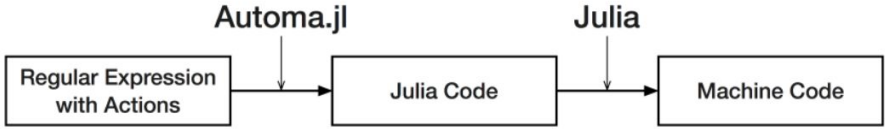
Note: Statistics collected on April 8, 2022

Programming Paradigms 1

Packages and resources that support various programming styles, Software Architecture and CS paradigms.

- Automata
- Control Flow
- Declarative Programming
  - Functional Programming
- DSL
- Grammatical Evolution
- Interpreters
- Language Comparison
- Macro
- Metaprogramming
  - Automatic Programming
- Multi Threading
- Polymorphism
  - Double Dispatch
  - Multiple Dispatch
- Program Analysis
- Reactive Programming
- STATIC ANALYSIS
  - Turnaround Time
- Style Guidelines

A Julia package for text validation, parsing, and tokenizing based on state machine compiler.



Examples:

- **Automa.jl :: A julia code generator for regular expressions - this package can do text validation, parsing, and tokenizing based on a state machine compiler.**

A tokenizer of octal, decimal, hexadecimal and floating point numbers

```
import Automa
import Automa.RegExp: @re_str
const re = Automa.RegExp

# Describe patterns in regular expression.
oct      = re"0o[0-7]+"
dec      = re"[-+]?[0-9]+"
hex      = re"0x[0-9A-Fa-f]+"
prefloat = re"[-+]?([0-9]+\.[0-9]*|[0-9]*\.[0-9]+)"
float    = prefloat | re.cat(prefloat | re"[-+]?[0-9]+", re"[eE][-+]?[0-9]+")
number   = oct | dec | hex | float
numbers  = re.cat(re.opt(number), re.rep(re" +" * number), re" *")
```

Programming Paradigms 2

Julia uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns.

```julia
# Use package and import desired positive/negative trait type aliases
using BinaryTraits
using BinaryTraits.Prefix: Can

# Define a trait and its interface contracts
@trait Fly
@implement Can{Fly} by fly(_, destination::Location, speed::Float64)

# Define your data type and implementation
struct Bird end
fly(::Bird, destination::Location, speed::Float64) = "Wohoo! Arrived! 🎉"

# Assign your data type to a trait
@assign Bird with Can{Fly}

# Verify that your implementation is correct
@check(Bird)

# Dispatch for all flying things
@traitfn flap(::Can{Fly}, freq::Float64) = "Flapping wings at $freq Hz"
```

Examples:

- **BinaryTraits.jl** :: easy-to-use trait library with formal interface specification support.
- **WhereTraits.jl** :: This package exports one powerful macro @traits with which you can extend Julia's where syntax.

- dispatch on functions returning Bool

```julia
@traits f(a) where {isodd(a)} = (a+1)/2
@traits f(a) where {!isodd(a)} = a/2
f(4) # 2.0
f(5) # 3.0
```

- dispatch on functions returning anything

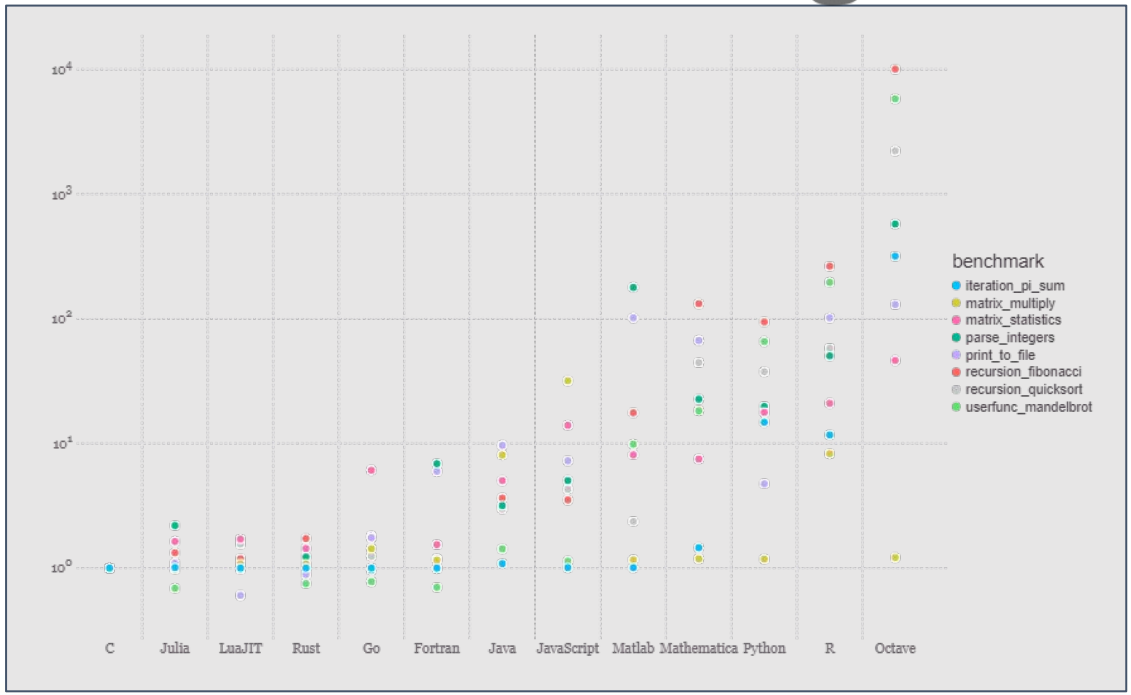
```julia
@traits g(a) where {Base.IteratorSize(a)::Base.HasShape} = 43
@traits g(a) = 1
g([1,2,3]) # 43
g(Iterators.repeated(1)) # 1
```

- dispatch on bounds on functions returning Types

```julia
@traits h(a) where {eltype(a) <: Number} = true
@traits h(a) = false
h([1.0]) # true
h([""]) # false
```

Performance

Julia Micro-Benchmarks



Performance

Language	Ease/readability	Lines of code	ST time	MT time
Numpy	Very good	15	6.8s	X
Basic Julia	Excellent	21	3.0s	0.6s
F2PY-Fortran	Very good	42	4.8s	1.3s

Performance

The Computer Language 22.03 Benchmarks Game

Julia costs versus Java costs

vs C vs Classic Fortran **vs Java**

vs Lisp vs Python

Always look at the source code.

If the fastest programs are hand-written vector instructions, does the host language matter? You might be more interested in the less optimised programs — more cpu seconds, less gz source code.

Concurrency

- Concurrency means the ability for a program to be decomposed into parts that can be run independently.
- Julia supports a variety of styles of concurrent computation.
    - A multithreaded computation(simultaneous work)
    - Distributed computing
    - GPU computing

LIVE DEMO

VIDEO

Any Questions?

Thank you for your time!

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC