### Julia tutorial for SF2524

### Giampaolo Mele

KTH Royal Institute of technology



SF2524

anonymous

## Why Julia?

"Julia has the performance of a statically compiled language like C, C++, etc while providing interactive dynamic behavior and productivity like Python, Matlab, LISP or Ruby."

— Bezanson et al, Julia: A Fast Dynamic Language for Technical Computing

## Why Julia?

"Julia has the performance of a statically compiled language like C, C++, etc while providing interactive dynamic behavior and productivity like Python, Matlab, LISP or Ruby."

— Bezanson et al, Julia: A Fast Dynamic Language for Technical Computing

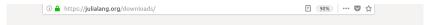
### Features and characteristics

- Mutiple dispatch
  - allows overloading of functions
  - call types determine what code is executed
  - Dynamic typing
    - "duck-typing"
    - allowing to indicate types
  - Just-in-time compilation
  - Moreover: easy parallel programming, easy high-precision computations, etc.

#### Technical information

- Julia: A Fresh Approach to Numerical Computing, J. Bezanson, A.
   Edelman, S. Karpinski and V. B. Shah (2017) SIAM Review, 59: 65-98
- Julia: A Fast Dynamic Language for Technical Computing, J.
   Bezanson, S. Karpinski, V. B. Shah and A. Edelman (2012) arXiv: 1209.5145
- Open source (MIT License)
   GitHub: https://github.com/JuliaLang/julia
- Current release 1.0.1 (October 2018)

# Install and run Julia (Ubuntu)



#### **Download Julia**

If you like Julia, please consider starring us on GitHub and spreading the word!



We provide several ways for you to run Julia:

- . In the terminal using the built-in Julia command line.
- In the browser on JuliaBox.com with Jupyter notebooks. No installation is required just point your browser there, login and start computing.
- JuliaPro by Julia Computing includes Julia and the Juno IDE, along with access to a curated set of packages for plotting, optimization, machine learning, databases and much more (requires registration).

#### Current stable release (v1.0.1)

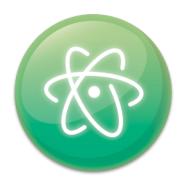
Windows Self-Extracting Archive (.exe) [help]	32-bit		64-bit	
fuerbl	Windows 7/Windows Server 2012 users also require TLS 'Essy Fix' update, and Windows Management Framework 3.0 or later			
macOS Package (.dmg) [help]	10.8+64-bit			
Generic Linux Binaries for x86 [help]	32-bit (GPG)		64-bit (GPG)	
Generic FreeBSD Binaries for x86 [help]	64-bit (GPG)			
Source	Tarball (GPG)	Tarball with dependencies (GPG) GitHub		GitHub

## Interactive usage

- (Ubuntu) open a terminal and execute the command "your\_path/julia-1.0.0/bin/julia"
- Getting help: type "? name\_of\_function"

### Which editor?

ATOM: https://atom.io/



- Editor+Console (interactive mode)
- Julia package: auto autocompletion, colors, formatting, etc
- Latex package: Greek letters, symbols, etc
- Many more package: git, other languages, etc
- Personalization

## Script example

### Create a file example\_script.jl

```
# comments what this script does
# define two numbers and sum them
a=2: b=\pi
c=a+b
# define two matrices and sum them
A=[ 1 2
   2 31
B=[ 2 -4
  -2 3 1
C = A + B
# define a function that sum of the elements of a matrix
function sum_matrix_el(M)
    m,n=size(M) # get the size of the matrix
    sumM=0
    for i=1:n
        for j=1:m
            sumM=sumM+M[i,j]
        end
    end
    return sumM
end
println("The sum of the el. of C is ",sum_matrix_el(C))
```

#### Hands on

#### We will now see

- Interactive usage
- Scripts
- Anonymous function
- Functions
- Load functions from a file
- multiple dispatch example

## Package systems

Julia is based on a package system. Many functions are available by installing and loading these packages.

#### Examples of packages:

- LinearAlgebra: norms, linear systems, eigenvalues, etc
- PyPlot, Plots: graphics, plots, etc
- Revise: software development
- BenchmarkTools: profiling
- Differential Equations, Julia FEM: solve differential equations
- Many more: ...

## Package systems

Julia is based on a package system. Many functions are available by installing and loading these packages.

#### Examples of packages:

- LinearAlgebra: norms, linear systems, eigenvalues, etc
- PyPlot, Plots: graphics, plots, etc
- Revise: software development
- BenchmarkTools: profiling
- Differential Equations, Julia FEM: solve differential equations
- Many more: ...
- Load a package: using name\_of\_package
- Download a package (two ways):
  - Package mode: press ] and then: add name\_of\_package
  - ► Load the Package manager: using Pkg; Pkg.add("name\_of\_package")

## Package systems

Julia is based on a package system. Many functions are available by installing and loading these packages.

#### Examples of packages:

- LinearAlgebra: norms, linear systems, eigenvalues, etc
- PyPlot, Plots: graphics, plots, etc
- Revise: software development
- BenchmarkTools: profiling
- Differential Equations, Julia FEM: solve differential equations
- Many more: ...
- Load a package: using name\_of\_package
- Download a package (two ways):
  - Package mode: press ] and then: add name\_of\_package
  - ► Load the Package manager: using Pkg; Pkg.add("name\_of\_package")

Hands on: LinearAlgebra package and matrix computation in Julia

#### **Factorize**

factorize(A)

Compute a convenient factorization of A, based upon the type of the input matrix. factorize checks A to see if it is symmetric/triangular/etc. if A is passed as a generic matrix. factorize checks every element of A to verify/rule out each property. It will short-circuit as soon as it can rule out symmetry/triangular structure. The return value can be reused for efficient solving of multiple systems. For example: A=factorize(A); x=A\b; y=A\cdot C.

Properties of A	type of factorization		
Positive-definite	Cholesky (see cholfact)		
Dense Symmetric/Hermitian	Bunch-Kaufman (see bkfact)		
Sparse Symmetric/Hermitian	LDLt (see ldltfact)		
Triangular	Triangular		
Diagonal	Diagonal		
Bidiagonal	Bidiagonal		
Tridiagonal	LU (see lufact)		
Symmetric real tridiagonal	LDLt (see ldltfact)		
General square	LU (see lufact)		
General non-square	QR (see qrfact)		

If factorize is called on a Hermitian positive-definite matrix, for instance, then factorize will return a Cholesky factorization.