



Scientific Computing and (Big) Data Analysis with Julia

Prof.Dr.Mehmet Hakan Satman
mhsatman@istanbul.edu.tr

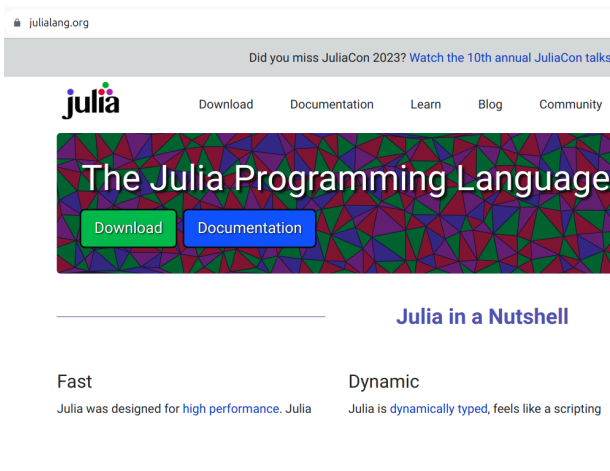
Istanbul University

2023.12.01



Julia Programming Language

julialang.org



The screenshot shows the homepage of the Julia Programming Language website. At the top, there is a navigation bar with the Julia logo and links for Download, Documentation, Learn, Blog, and Community. Below the navigation bar is a large banner with a colorful, abstract geometric pattern. The banner contains the text "The Julia Programming Language" and two buttons: "Download" and "Documentation". Below the banner, there is a section titled "Julia in a Nutshell" which is divided into two columns. The left column is titled "Fast" and contains the text "Julia was designed for high performance. Julia". The right column is titled "Dynamic" and contains the text "Julia is dynamically typed, feels like a scripting". In the bottom right corner of the website, there is a small image of a classical building.

julialang.org

Did you miss JuliaCon 2023? [Watch the 10th annual JuliaCon talks](#)

Download Documentation Learn Blog Community

The Julia Programming Language

Download Documentation

Julia in a Nutshell

Fast
Julia was designed for high performance. Julia

Dynamic
Julia is dynamically typed, feels like a scripting

Julia Programming Language

High Performance Computing

- Julia was designed for high performance. Julia programs automatically compile to efficient native code via LLVM, and support multiple platforms (Windows, MacOS, Linux, etc.)¹.

¹<https://julialang.org/>



Julia Programming Language

Dynamic

- Julia is dynamically typed, feels like a scripting language, and has good support for interactive use, but can also optionally be separately compiled².

²<https://julialang.org/>



Julia Programming Language

Composable

- Julia uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns. The talk on the Unreasonable Effectiveness of Multiple Dispatch explains why it works so well³.

³<https://julialang.org/>



Julia Programming Language

Open Source

- Julia is an open source project with over 1,000 contributors. It is made available under the MIT license. The source code is available on GitHub⁴.

⁴<https://julialang.org/>



Julia Programming Language

First things first!

```
println("Hello , world")
```

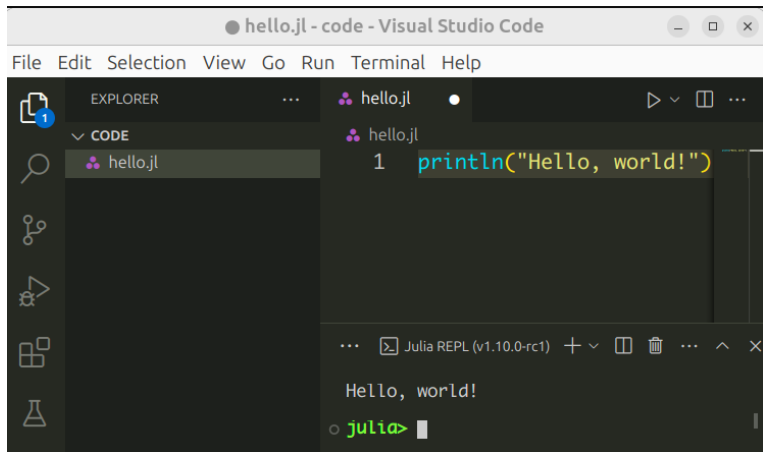


Welcome!



Julia Programming Language

The editor: Visual Studio Code



The screenshot shows the Visual Studio Code interface with a file named `hello.jl` open. The Explorer sidebar on the left shows the file structure. The main editor area displays the code `println("Hello, world!")` on line 1. The bottom panel shows the Julia REPL (v1.10.0-rc1) with the output `Hello, world!` and the prompt `julia>`.



Linear Regression

The formulation

$$y = \beta_0 + \beta_1 x + \varepsilon \quad (1)$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \quad (2)$$

$$\hat{\beta} = (X'X)^{-1}X'y \quad (3)$$



Linear Regression

Sample Data

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}, y = \begin{bmatrix} 2 \\ 5 \\ 5 \\ 8 \\ 12 \end{bmatrix} \quad (4)$$



Linear Regression

The Matrix Solution

```
using LinearAlgebra

x = [1, 2, 3, 4, 5]
y = [2, 5, 5, 8, 12]
X = hcat(ones(5), x)
betahats = inv(X'X)X'y
println(betahats)
```

```
julia> include("reg-matrix.jl")
[-0.5, 2.3]
```



Linear Regression

Pseudo Inverse - Numerical Fit

```
x = [1, 2, 3, 4, 5]
y = [2, 5, 5, 8, 12]

betahats = hcat(ones(5), x) \ y
println(betahats)
```

```
julia> include("reg-simple.jl")
[-0.5, 2.3]
```



Linear Regression

The GLM package

```
using GLM

x = [1, 2, 3, 4, 5]
y = [2, 5, 5, 8, 12]

result = lm(hcat(ones(5), x), y)

println(result)
```



Linear Regression

The GLM package - Results

```
julia> include("reg-glm.jl")
```

Coefficients:

	Coef.	Std. Error	t	Pr(> t)	Lower 95%	Upper 95%
x1	-0.5	1.25565	-0.40	0.7171	-4.49605	3.49605
x2	2.3	0.378594	6.08	0.0090	1.09515	3.50485

```
julia> GLM.r2(result)
```

0.9248251748251748



XOR

eXclusive OR

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

Table: $y = \text{xor}(x_1, x_2)$



Symbolic Regression

```
using SymbolicRegression, MLJ

x = (
  x1 = Float64[1, 1, 0, 0],
  x2 = Float64[1, 0, 1, 0]
)

y = Float64[0, 1, 1, 0]
```



Symbolic Regression

```
model = SRRegressor(  
    iterations = 50,  
    binary_operators = [+ , - , *],  
    unary_operators = [abs],  
    should_simplify = true,  
    save_to_file = false)
```



Symbolic Regression

```
mach = machine(model, x, y)
fit!(mach)
report(mach)
@info predict(mach, x)
```



Symbolic Regression

Hall of Fame:

```
-----  
Complexity    Loss          Score      Equation  
1             2.500e-01    3.604e+01    y = 0.5  
4             0.000e+00    1.201e+01    y = abs(x1 - x2)  
-----  
[ Info: [0.0, 1.0, 1.0, 0.0]
```

