desafio 15

Sophia ra277230

2025-11-16

Instalação de Júlia

```
#install.packages("JuliaCall")
library(JuliaCall)
```

```
## Warning: pacote 'JuliaCall' foi compilado no R versão 4.5.2
```

```
julia <- julia_setup()
```

```
## Julia version 1.12.1 at location C:\Users\Shophia\AppData\Local\Programs\JULIA-~1.1\bin will be used
```

```
## Loading setup script for JuliaCall...
```

```
## Finish loading setup script for JuliaCall.
```

```
system("where julia", intern = TRUE)
```

```
## [1] "C:\\Users\\Shophia\\AppData\\Local\\Programs\\Julia-1.12.1\\bin\\julia.exe"
```

```
# Instala BenchmarkTools
julia_command('import Pkg; Pkg.add("BenchmarkTools")')
```

```
# Se quiser, pode instalar outros pacotes necessários também
julia_command('import Pkg; Pkg.add("Random")')
```

```
library(JuliaCall)

# Conecta com Julia
julia <- julia_setup()  # já funciona, como você confirmou

# Carrega pacotes Julia necessários
julia_command("using BenchmarkTools, Distributed, Random, Base.Threads")
```

```
# Definindo tamanho do vetor
julia_command("N = 10^7; data = rand(N)")
```

```
## 10000000-element Vector{Float64}:
##  0.013287370268120435
##  0.9479779180083018
##  0.15962547553448025
##  0.7621101455466226
##  0.7131904993199125
##  0.8259096116209588
##  0.23678106403454124
##  0.48055473904825685
##  0.9820316688158959
##  0.5572966777214305
##
##  0.07062202617505009
##  0.9683159933576634
##  0.09478501162803676
##  0.07209223728735026
##  0.6914483634495989
##  0.8490735950438573
##  0.9185705369217496
##  0.5049100716481393
##  0.21247693046308003
```

```
# Função em série
julia_command("
function loop_serial(data)
    result = zeros(length(data))
    for i in 1:length(data)
        result[i] = sqrt(data[i]) * sin(data[i])
    end
    return result
end
")
```

```
## loop_serial (generic function with 1 method)
```

```
# Função paralela usando Threads
julia_command("
function loop_parallel(data)
    result = zeros(length(data))
    Threads.@threads for i in 1:length(data)
        result[i] = sqrt(data[i]) * sin(data[i])
    end
    return result
end
")
```

```
## loop_parallel (generic function with 1 method)
```

```
# Medindo tempo série
julia_command("println(\"Tempo série:\"); @time result_serial = loop_serial(data)")
```

```
## 10000000-element Vector{Float64}:
```

```
##  0.0015316014768684697
##  0.7908283276406304
##  0.06350492566824868
##  0.6027548874008817
##  0.5525180971556928
##  0.6681145144464765
##  0.11414442400158388
##  0.3204558671475393
##  0.8241219958392413
##  0.3948314594180875
##
##  0.018752069115615853
##  0.8107747714953846
##  0.02913795100602007
##  0.01934000124010465
##  0.5302304495388184
##  0.6917050696813656
##  0.7616904167073839
##  0.34372324551802014
##  0.09720641690731588
```

```
# Medindo tempo paralelo
julia_command("println(\"Tempo paralelo:\"); @time result_parallel = loop_parallel(data)")
```

```
## 10000000-element Vector{Float64}:
##  0.0015316014768684697
##  0.7908283276406304
##  0.06350492566824868
##  0.6027548874008817
##  0.5525180971556928
##  0.6681145144464765
##  0.11414442400158388
##  0.3204558671475393
##  0.8241219958392413
##  0.3948314594180875
##
##  0.018752069115615853
##  0.8107747714953846
##  0.02913795100602007
##  0.01934000124010465
##  0.5302304495388184
##  0.6917050696813656
##  0.7616904167073839
##  0.34372324551802014
##  0.09720641690731588
```