ARTIGO
DOI: https://doi.org/10.21728/p2p.2024v11n1e-7060
Data de submissão: 27/05/2024 Data de aprovação: 11/08/2024 Data de publicação: 13/08/2024

LINGUAGEM DE PROGRAMAÇÃO JULIA
uma linguagem feita para a ciência

Gabriel dos Santos Lima[1]
Instituto Federal de Sergipe
gabriel_s.lima@outlook.com

José Aprígio Carneiro Neto[2]
Instituto Federal de Sergipe
jose.neto@ifs.edu.br

Resumo

Na ciência da computação, são prevalentes diversas línguas de programação para solução de problemas interdisciplinares, seja o *Matlab* para cálculo numérico, *R* para estatística, análise de dados com *Python*, entre outras. No entanto, neste contexto, existe uma curva de aprendizado associada ao domínio da linguagem de programação utilizada, portanto, um projeto que englobe duas ou mais ferramentas distintas no seu fluxo de operação tende a requerer um maior aprendizado e esforço de integração. Como alternativa, a linguagem de programação Julia oferece uma saída prática, disponibilizando uma série de ricas bibliotecas focadas no uso científico da computação nos mais diversos domínios do conhecimento, permitindo o uso de uma única linguagem de programação para o todo de um projeto. Diante desse contexto, este artigo teve por objetivo fazer uma contextualização da linguagem de programação Julia, mostrando o seu ecossistema e identificando os principais diferenciais que a linguagem possui em relação as demais linguagens utilizadas. Além disso, esta pesquisa teve a oportunidade de explorar algumas aplicações científicas que demonstram a capacidade da linguagem de programação Julia na prática, bem como explorar as suas vantagens e desvantagens.

Palavras-chave: Julia; ciência da computação; linguagens de programação.

JULIA PROGRAMMING LANGUAGE
a language made for science

Abstract

In computer science, several programming languages are prevalent for solving interdisciplinary problems, be it Matlab for numerical calculation, R for statistics, data analysis with Python, among others. However, in this context, there is a learning curve associated with mastering the programming language used, therefore, a project that encompasses two or more different tools in its operation flow tends to require greater learning and integration effort. As an alternative, the Julia programming language offers a practical solution, providing a series of rich libraries focused on the scientific use of computing in the most diverse domains of knowledge, allowing the use of a single programming language for the entire project. Given this context, this article aimed to contextualize the Julia programming language, showing its ecosystem and identifying the main differences that the language has in relation to other languages used. Furthermore, this research had the opportunity to explore some scientific applications that demonstrate the capabilities of the Julia programming language in practice, as well as explore its advantages and disadvantages.

[1] Graduando em Ciência da Computação pelo Instituto Federal de Sergipe – IFS (2024).

[2] Pós-Doutor em Engenharia e Computação Inteligente pelo Instituto Politécnico do Porto – ISEP/IPP, em Porto, Portugal (2024). Pós-Doutor em Engenharia de Produção e Sistemas pela Universidade do Minho – UNIMINHO, em Braga, Portugal (2023). Pós-Doutor em Ciência da Computação pela Universidade Federal de Sergipe - UFS (2022). Doutor em Ciência da Propriedade Intelectual pela Universidade Federal de Sergipe – UFS (2018). Mestre em Engenharia de Software pelo Centro de Estudos e Sistemas Avançados do Recife – C.E.S.A.R. EDU (2013). Especialista em Tecnologias da Informação, com ênfase em Cliente/Servidor, pela Universidade Federal do Ceará – UFC (2001). Graduado em Formação Pedagógica em Informática pelo Centro Universitário Leonardo Da Vinci – UNIASSELVI (2020). Graduado em Processamento de Dados pela Universidade Estadual do Piauí – UESPI (1997).

Esta obra está licenciada sob uma licença Creative Commons Attribution 4.0 International (CC BY-NC-SA 4.0).

Keywords: Julia; computer science; programming languages.

LENGUAJE DE PROGRAMACIÓN JULIA
un lenguaje hecho para la ciencia

Resumen

En informática prevalecen varios lenguajes de programación para la resolución de problemas interdisciplinarios, ya sea Matlab para cálculo numérico, R para estadística, análisis de datos con Python, entre otros. Sin embargo, en este contexto, existe una curva de aprendizaje asociada al dominio del lenguaje de programación utilizado, por lo que un proyecto que engloba dos o más herramientas diferentes en su flujo de operación tiende a requerir un mayor esfuerzo de aprendizaje e integración. Como alternativa, el lenguaje de programación Julia ofrece una solución práctica, proporcionando una serie de ricas bibliotecas enfocadas al uso científico de la informática en los más diversos dominios del conocimiento, permitiendo el uso de un único lenguaje de programación para todo el proyecto. Ante este contexto, este artículo tuvo como objetivo contextualizar el lenguaje de programación Julia, mostrando su ecosistema e identificando las principales diferencias que tiene el lenguaje con relación a otros lenguajes utilizados. Además, esta investigación tuvo la oportunidad de explorar algunas aplicaciones científicas que demuestran las capacidades del lenguaje de programación Julia en la práctica, así como explorar sus ventajas y desventajas.

Palabras clave: Julia; ciencia de la computación; lenguajes de programación.

1 INTRODUÇÃO

A linguagem de programação Julia[3] teve sua origem em 2012 e foi anunciada publicamente na entrada do *blog* "*Why we created Julia*" (Bezanson *et al.*, 2012), onde os principais idealizadores, Jeff Bezanson, Stefan Karpinski, Viral B. Shah e Alan Edelman discutiram sua intenção para a linguagem, além disso, nessa mesma publicação, delimitaram a ambição que o projeto trazia consigo, cujo objetivo primário era de desenvolver a LP[4] ideal para computação científica.

Coletivamente, o time responsável por criar a linguagem Julia, em 2012, era composto por desenvolvedores de diversas áreas de atuação e com extensa experiência em linguagens de programação de uso científico. É descrito no parágrafo inicial do supracitado *blog* que os desenvolvedores do projeto, ao longo de suas carreiras, utilizaram ferramentas como *Matlab*, *Mathematica*, *R*, linguagens como *Python*, *Ruby*, *Perl*, *Java* e *C* extensivamente, ao ponto de ganhar dimensão dos inconvenientes que afetavam o seu fluxo de trabalho, tal como performance no caso de *Python*, a quantidade de *boilerplate*[5] em *Java* ou a complexidade do gerenciamento de memória em *C*, ou seja, essa experiência de uso permitiu que uma visão fosse construída para Julia, a construção de uma LP que sanasse os incômodos das outras linguagens de programação no cenário da performance e da ciência. Em suma, pode-se inferir que foi iniciado ali um projeto muito ambicioso (Bezanson *et al.*, 2012).

Após um longo período de desenvolvimento, em 2018, Julia alcançou sua versão estável 1.0, tal marco alcançou o objetivo inicial do projeto, disponibilizando para a comunidade científica uma LP unificada para o desenvolvimento de bibliotecas especializadas para suas aplicações, favorecendo que estas tenham entre si a menor barreira possível na integração dos diversos componentes que compõem um sistema complexo, além de entregar na promessa de combinar os melhores aspectos de linguagens como *Python*, *Ruby*, *C*, *Perl*, dentre outras em uma única sintaxe, tornando Julia uma das LPs mais únicas para o desenvolvimento científico da atualidade.

Como descrito na publicação *Open Source Julia Data Science* (Storopoli; Huijzer; Alonso, 2021), o grande benefício de Julia se encontra na solução do problema das duas linguagens, onde um programador pode optar por modelar um algoritmo de algum tipo em uma LP de sintaxe fácil, na premissa de reescrever o algoritmo em uma linguagem de alta

[3] Disponível em: https://julialang.org/downloads/.

[4] Acrônimo para Linguagem de Programação ou Língua de Programação.

[5] Definido como código verboso, que é necessário se escrever muito para obter pouca funcionalidade.

performance. Dessa forma, o trabalho cumulativo se encontra dobrado em relação a quantidade de código que necessita ser desenvolvido, no entanto, como Julia oferece tanto a performance e praticidade em sua sintaxe, nota-se que sua adoção permite que o código seja modelado e polido sem a necessidade da codificação em uma nova linguagem de programação.

Atualmente, Julia se mostra uma LP extremamente versátil, atuando com destaque nos campos da ciência de dados, aprendizado de máquina, programação paralela, visualização e plotagem de dados, cálculo numérico, computação gráfica, além de ser parte de diversos projetos de biologia, química e física computacional, tudo possível devido ao trabalho arquitetônico da LP ao oferecer bibliotecas nativas, planejadas da concepção para serem funcionalmente completas, removendo a necessidade de boa parte das intervenções do desenvolvedor e, além disso, também feitas para que a integração com as demais bibliotecas que um determinado projeto possa englobar seja facilitada. Tudo isso é impulsionado para frente por uma comunidade engajada de cientistas que contribuem com o desenvolvimento de outras extensões que especializam ainda mais a linguagem e seu ecossistema (Tomasi; Giordano, 2018).

A Linguagem Julia apresenta uma pequena parcela do mercado quando comparada a outras linguagens de programação da ciência de dados, tal como *Python*, *R*, entre outras (StackOverflow Survey, 2022) e, sendo Julia uma linguagem desenvolvida como substituto moderno dessas linguagens, vemos então uma grande oportunidade de crescimento em número de programadores que utilizam Julia no cotidiano. Considerando o curto espaço de tempo desde o lançamento da versão 1.0 de Julia, é natural que a LP não possua uma grande presença no dia a dia de muitos desenvolvedores após, apenas, 4 anos de ampla disponibilidade, portanto, para maior disseminação do que se trata essa linguagem de programação, procura-se desenvolver uma produção textual informativa e comparativa sobre Julia, para elucidar o público acadêmico sobre as possibilidades e oportunidades que o uso de Julia fornece.

Como valia ao presente artigo, observa-se que é do interesse do coletivo da ciência da computação que tecnologias inovadoras sejam exploradas e, para alcançar tal finalidade, a composição deste artigo terá foco em destacar o cenário geral da linguagem de programação Julia, em trazer à tona aplicações reais que mostram o potencial da linguagem na ciência e seu benefício para a ciência da computação como um todo.

Portanto, o objetivo deste artigo foi fazer uma contextualização da linguagem de programação Julia, mostrando o seu ecossistema e identificando os principais diferenciais que a linguagem possui em relação as demais linguagens utilizadas no cotidiano de programação. Além disso, esta pesquisa teve a oportunidade de explorar algumas aplicações científicas que

demonstram a capacidade da linguagem de programação Julia na prática, bem como explorar suas vantagens e desvantagens.

2 METODOLOGIA

A metodologia utilizada neste artigo utiliza procedimentos de pesquisa bibliográfica e documental, atentando-se principalmente às especificações da linguagem, material produzido pelos autores de Julia e aos projetos científicos elaborados com esta linguagem. O objetivo é exploratório, o qual consiste na exploração da linguagem Julia e suas aplicações para alcançar uma visão abrangente de suas características. No tópico da abordagem, a pesquisa enquadra-se como descritiva, detalhando características funcionais de Julia e sua participação na produção científica na atualidade, tomando como base a documentação da linguagem e projetos já existentes e utilizando, para esse fim, uma visão longitudinal do tempo em que Julia alcançou sua versão 1.0 até o presente momento.

Os benefícios propostos pelo presente artigo se encontram no potencial da disseminação do conhecimento de uma nova linguagem e suas aplicações na produção científica. Julia, no seu estado atual, vem ganhando momento como alternativa nos usos científicos da programação, no entanto, de forma ampla, seus benefícios e funcionalidades permanecem com alcance popular limitado, logo ao explorar e exaltar os principais diferenciais que a linguagem possui, procura-se expor os avanços propostos pela linguagem de programação, as vantagens performáticas que poderão ser obtidas em projetos, além de comunicar melhorias no processo da escrita e manutenção do código que a linguagem pode trazer para o meio científico.

3 ANÁLISE E DISCUSSÃO DOS RESULTADOS

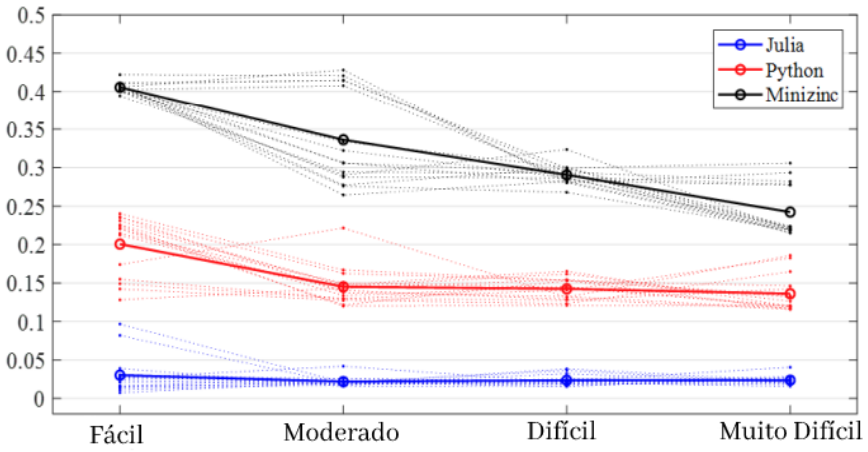
3.1 FUNDAMENTOS DE JULIA

O *design* arquitetônico de Julia representa uma quebra de paradigma no cenário geral da programação em diversos aspectos de sua funcionalidade. O primeiro fator diferencial é o fato da linguagem ter sido desenvolvida com sua atuação científica imperativa no processo de desenvolvimento. Embora Julia possua sua utilização como linguagem de programação geral, sua concepção teve como objetivo fornecer performance para aplicações tidas como intensivas em computação e, para isso, diversas medidas foram tomadas para garantir a praticidade de uso da LP, tal como a existência de macros nativos na linguagem que permitem ao desenvolvedor a automatização de medidas para incremento de performance, assim como, permite que o

benchmarking[6] seja facilitado através de macros como *@time*, *@timev* e *@btime*, que detalham a performance em função do tempo e de ocupação de memória em sua execução (Bezanson *et al.*, 2021). Além disso, a sintaxe para criação de macros é simples e eficiente, permitindo que desenvolvedores criem suas próprias utilidades sem impedimentos.

Aplicando essa performance em problemas cotidiano, Julia também demonstra vantagens quando comparada a soluções tradicionais, visível na obra de Bukhari (2022), que compara entre *Python*, Julia e *Minizinc*, o tempo (em segundos) de computação necessário para resolver 14 *puzzles* nos nível de complexidade Fácil, Moderado, Difícil e Muito Difícil, como mostra a Figura 1.

Figura 1 – Comparação de tempo de computação entre Julia, *Python* e *Minizinc*.



Fonte: Adaptado de Bukhari *et al.*, 2022.

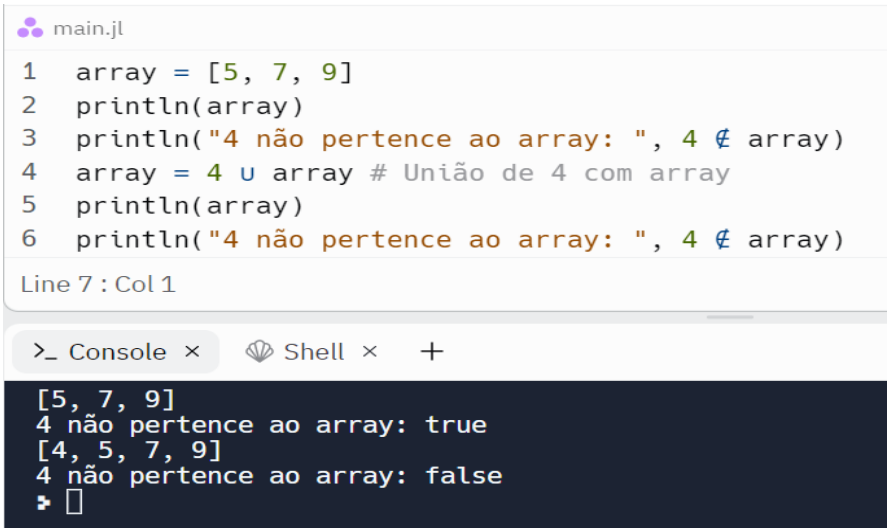
O segundo fator para seu destaque é a forma como a metaprogramação pode ser utilizada na linguagem, funcionalidade que teve suas origens na linguagem de programação *Lisp*, onde variáveis de programa podem receber expressões como valor, pois expressões também são um tipo de dado dentro de Julia (*Expr*), possibilitando que essas expressões sejam interpretadas no código e, potencialmente, interpoladas entre si no código ou utilizando macros, permitindo que programas possam ser dinamicamente construídos em tempo de execução, combinando e avaliando expressões de acordo com a entrada de dados da aplicação. (Boudreau, 2020)

Outro aspecto interessante da linguagem Julia, especialmente útil para as ciências matemáticas, é o suporte a elementos *Unicode* integrados ao código, que permite a utilização de símbolos nas suas respectivas funções esperadas de forma completamente natural. Como

[6] Processo sistemático para medida de determinado atributo de um objeto de análise em comparação a outros objetos correlatos.

exemplo, podemos utilizar o símbolo "∉" (não pertence) para realizar uma lógica em um *array*, para determinar se um elemento não pertence àquele *array*, retornando "*verdadeiro*" ou "*falso*" dessa inspeção (Le, 2021). Dessa forma, a linguagem se aproxima na proporção de "um para um entre código e matemática"[7] (Storopoli; Huijzer; Alonso, 2021), como pode ser observado na Figura 2.

Figura 2 – Demonstração da codificação matemática em Julia.



Fonte: Autoria própria.

Noutra particularidade, identifica-se que a implementação de Julia em despachos múltiplos (recurso comum de outras linguagens da programação) aproveita o melhor das demais implementações para oferecer uma forma de polimorfismo mais eficiente. A princípio, é notável a possibilidade da configuração de funções padrões que aceitam qualquer tipo de argumento (*::Any*) e qualquer quantidade de argumentos, graças ao operador *splat* suportado pela linguagem, que pode ser atrelado a tipos de dados (Exemplo: I::*Integer*...). Além destes fatores, Julia verifica cada tipo dos argumentos fornecidos a função, para garantir a ambiguidades e que a chamada para uma determinada função seja a correta. Por fim, as definições de funções aceitam ainda supertipos, que englobam outros subtipos, por exemplo, *Real*, que engloba valores inteiros e de pontos flutuantes. (Arslan *et al*, 2023)

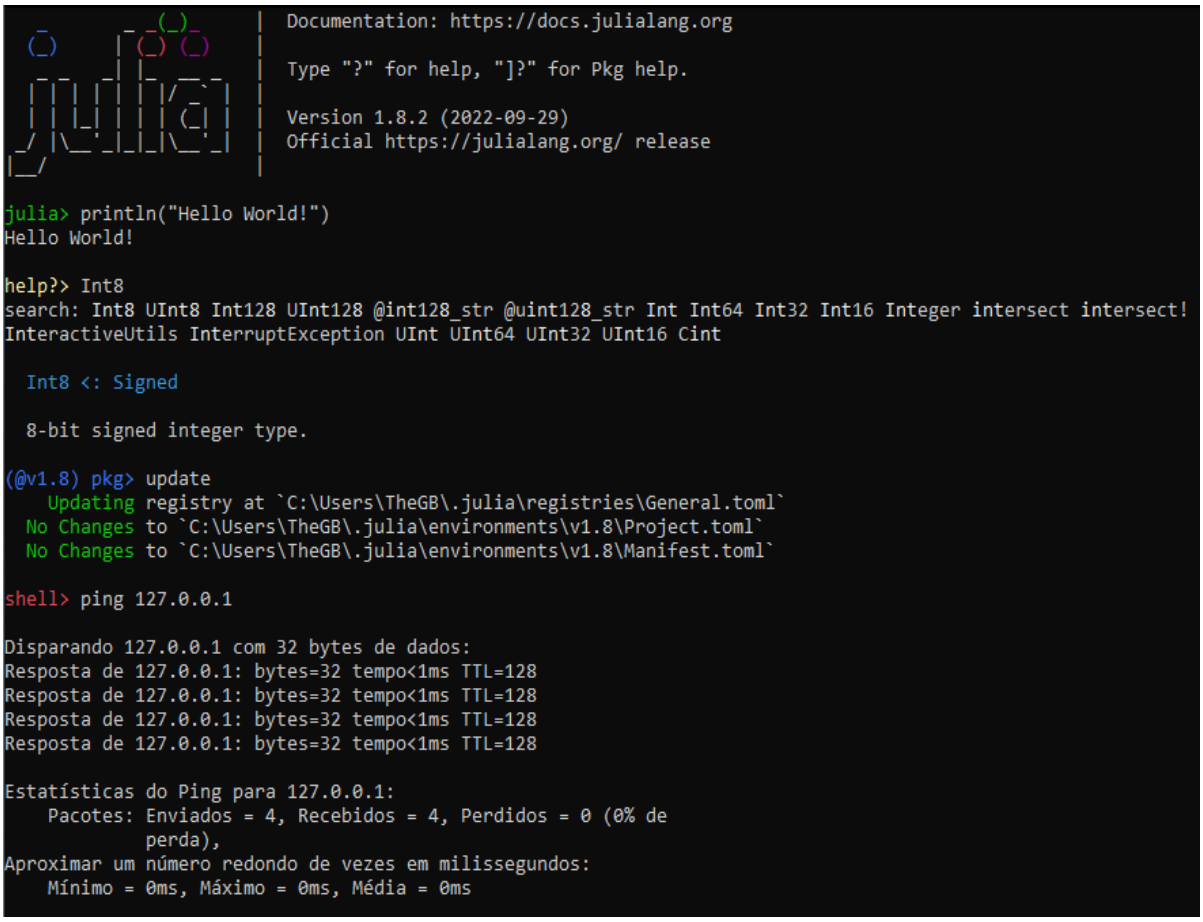
Adicionalmente, Julia possui um ambiente *REPL*[8] bem estruturado e acolhedor, possibilitando a execução e manipulação de códigos de Julia dos usuários, assim como, fornece

[7] A intencionalidade desse *design* é elucidada por Alan Edelman (cocriador de Julia) em sua *TED talk* no *MIT* "*A programming language to heal the planet together: Julia*": https://youtu.be/qGW0GT1rCvs.

[8] Acrônimo para *Read-eval-print loop*. Aplicações que leem e reagem a entradas de usuário em loop contínuo.

acesso a documentação de Julia em seu modo ajuda (*help*), que é acessível via tecla "**?**", sem nenhum *input* de texto anterior. Além do modo ajuda, o *REPL* de Julia permite acessar o modo de gerenciamento de pacotes (via tecla "**]**") e o modo *shell* (via tecla "**;**"), permitindo que comandos da *shell* nativa do sistema operacional sejam executados sem que seja necessário sair da linha de comando atual, como mostra a Figura 3.

Figura 3 – Demonstração da *REPL* de Julia.



Fonte: Autoria própria.

A linguagem de programa Julia oferece ao desenvolvedor um ecossistema repleto de bibliotecas para diversas funcionalidades, que podem ser facilmente adicionadas através do seu gerenciador de pacotes, isto se as bibliotecas nativas já não forem satisfatórias para a resolução do problema em questão. Dentre as funcionalidades mais reconhecidas, observa-se as bibliotecas nativas para estatística (*Statistics.jl*), para problemas envolvendo grafos (*Graphs.jl*), Algebra Linear (*LinearAlgebra.jl*), Encadeamento (*Chain.jl*), dentre muitas outras. Quando se

trata de bibliotecas externas, as mais proeminentes são: *DifferentialEquations.jl*[9], para solução de equações diferenciais; o *ModelingToolkit.jl*[10], para elaboração de modelos de alta performance de aprendizado de máquina; o *Genie.jl*[11], para construção aplicações na *web*; e o *CUDA.jl*[12], para realização de interface com *GPUs* modernas. (Le, 2021)

3.2 APLICAÇÕES DE JULIA

Dentro da esfera científica, Julia se destaca naturalmente devido à sua orientação à ciência no cerne de seu *design*, de forma que sua aplicação em projetos se tornou acentuada nos últimos anos, desde sua versão estável 1.0, assim como, obteve adoção ampla de diversas empresas, aumentando seu alcance de forma considerável, embora, ainda não seja uma tendência dominante. (Sobyte, 2021)

Essa popularidade na comunidade científica é corroborada por índices de popularidade como *IEEE Spectrum*[13] e pela pesquisa anual do *Stack Overflow*[14] da comunidade, nas quais Julia apresenta uma presença forte de publicações no agregador acadêmico *IEEE Xplore*, uma quantidade elevada de repositórios ativos no *Github* e, pela primeira vez, participou do *top* 5 das linguagens de programação mais amadas do *Stack Overflow* de 2022 (Kilpatrick, 2022).

Estendendo o olhar para os projetos que Julia habilita, pode-se notar uma ampla gama de projetos nos diversos domínios da ciência da computação, em paralelo com distintos núcleos de conhecimento, tal como ciências da natureza, logística, matemática, engenharia, entre outros campos de estudo.

Elencando exemplos, o primeiro projeto notável a ser destacado é o *SatelliteToolbox.jl*[15] (Chagas *et al.*, 2018), projeto brasileiro desenvolvido internamente no INPE[16], para atuar como pacote de funcionalidades para satélites em suas soluções de simulação de percurso, análise de órbitas, propagação orbital, entre outras funções, de tal maneira que Julia se estabelece como uma peça ativa no processo de pesquisa espacial no Brasil. (Chagas, 2019)

[9] Disponível em: https://github.com/SciML/DifferentialEquations.jl.

[10] Disponível em: https://github.com/SciML/ModelingToolkit.jl.

[11] Disponível em: https://github.com/GenieFramework/Genie.jl.

[12] Disponível em: https://github.com/JuliaGPU/CUDA.jl.

[13] Disponível em: https://spectrum.ieee.org/top-programming-languages/.

[14] Disponível em: https://survey.stackoverflow.co/2022/.

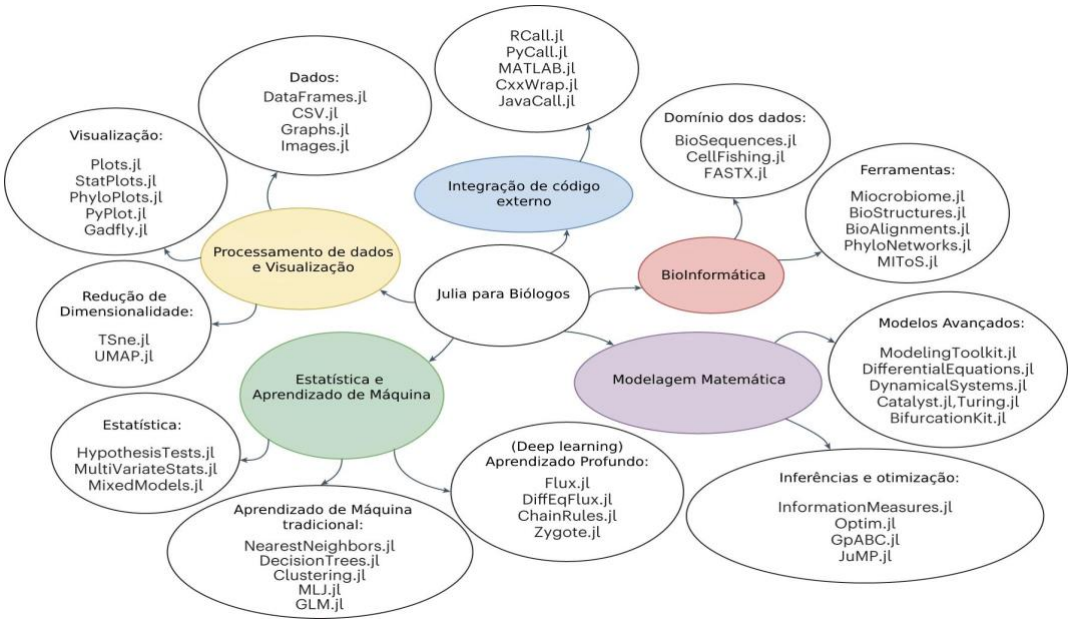
[15] Disponível em: https://github.com/JuliaSpace/SatelliteToolbox.jl.

[16] Acrônimo do órgão nacional, Instituto Nacional de Pesquisas Espaciais.

De acordo com o arquivo de casos de uso da fundação *Julia Computing*[17], Julia já ofereceu integração facilitada para diversas modalidades de conhecimento em aplicações práticas da linguagem, incluindo aplicações nos campos da Medicina, Ecologia, Robótica, Macroeconomia, ente outras. Dentre os destaques de sua eficácia, o portal exalta projetos como a aplicação de Julia dentro do *MIT Robot Locomotion Group*, que substituiu componentes escritos em *C++*, nos seus modelos de locomoção robótica, por novas implementações em Julia[18], além disso o portal descreve ainda de que forma a linguagem Julia pôde aumentar a performance da segurança de redes da empresa Cisco em 7,5x (sete vezes), com apenas duas linhas de código[19], além de contar com um caso de uso brasileiro no BNDES, no qual uma série de sistemas financeiros tiveram uma melhoria de 10x (dez vezes) com sua nova implementação em Julia[20].

Para melhor representar a riqueza apresentada pelo ecossistema de Julia quando voltada à ciência computacional, é possível organizar em forma de mapa mental uma série de bibliotecas que, quando associadas, podem auxiliar em tarefas complexas de certo domínio do conhecimento. Na Figura 4, observa-se como o campo da Bio Computação classifica e usufrui da linguagem de programa Julia.

Figura 4 – Julia para Biólogos.



Fonte: Adaptado de Roesch *et al.*, 2023.

[17] Disponível em: https://juliahub.com/case-studies/.

[18] Disponível em: https://juliahub.com/case-studies/mit-robotics/.

[19] Disponível em: https://juliahub.com/case-studies/cisco/.

[20] Disponível em: https://juliahub.com/case-studies/bndb/.

Em suma, a participação de Julia em seu curto tempo de maturidade se mostra significativa, de tal maneira que sua contribuição para a ciência pode ser potencializada conforme seu alcance na produção científica aumente para outros públicos.

3.3 ENSINO DE PROGRAMAÇÃO COM JULIA

Cabe ressalvar, a análise de Julia como uma ferramenta didática inicial de codificação, mostrando os benefícios que a adoção da linguagem pode trazer quando comparada com a usual medida de ensino de pseudocódigo para ingressantes no campo da programação.

Comparativamente, Julia, assim como pseudocódigo, suprime a necessidade do uso de chaves ({ }) para organização do código em blocos funcionais, graças ao uso da palavra reservada *end*, que denota o final de execução de uma função, condicional ou *loop* (Kilpatrick, 2021), algo que também pode ser percebido em pseudocódigo, como observado em *Portugol* com os comandos *fimse*, *fimpara* e *fimcaso*. Essas pontuações de encerramento de um determinado trecho de código, ajudam os desenvolvedores iniciantes a delimitarem suas funcionalidades com maior praticidade, sem lidar com uma série de chaves de abertura e fechamento em funções ou *loops* com muitos componentes internos. (Leite *et al.*, 2013)

Como outro ponto notório da linguagem, seu índice inicial para posições é 1, diferentemente da grande maioria das demais linguagens de programação, que possuem índice 0. Essa escolha foi feita devido a inspiração da linguagem em *Lua*, *Mathematica*, *Matlab* e *R*, tais linguagens que também usam o índice 1 para favorecer uma lógica numérica mais intuitiva, de tal forma que, para um programador iniciante, apresenta uma vantagem ao habilitar uma experiência de codificação mais intuitiva. (Kilpatrick, 2021)

Por fim, para os programadores mais curiosos, Julia apresenta macros que expõem, etapa por etapa, a forma como ocorre a geração de código para nível de máquina, sem a necessidade de outra aplicação intermediária, sendo assim possível compreender melhor a maneira de como a codificação realizada será convertida para código de baixo nível. Para utilização dessas macros, basta que sejam utilizados os códigos *@code_lowered*, *@code_typed*, *@code_llvm* ou *@code_native*, antes da chamada de um método ou expressão para que sua representação em cada uma das etapas de conversão seja exibida (Velho *et al.*, 2022), como pode ser observado na Figura 5.

Figura 5 – Demonstração do código nativo gerado na soma de dois *arrays* de números.

```
julia> @code_native [5, 3, 1] + [6, 1, 9]
        .text
        .file   "+"
        .globl  "julia_+_44"                    # -- Begin function julia_+_44
        .p2align        4, 0x90
        .type   "julia_+_44",@function
"julia_+_44":                           # @"julia_+_44"
; ┌ @ arraymath.jl:12 within `+`
        .cfi_startproc
# %bb.0:                                # %top
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        pushq   %r15
        pushq   %r14
        pushq   %r13
        pushq   %r12
        pushq   %rbx
        andq    $-32, %rsp
```

Fonte: Autoria própria.

3.3 DESVANTAGENS ATUAIS NA ADOÇÃO DE JULIA

Quando comparada às demais linguagens de programação já estabelecidas no meio da computação científica (*Python*, *R*, entre outras), Julia, com apenas 4 anos desde sua introdução no mercado como uma linguagem estável e com um público proporcionalmente menor do que as demais LPs com décadas de atividade, demonstra um ecossistema significativamente menor, considerando principalmente os seguintes fatores: falta de ferramentas específicas para o auxílio no desenvolvimento de código em Julia; quantitativo de pacotes publicados inferior as LPs com maior maturidade; e o fato de que certas áreas de pesquisa ainda não possuem muita familiaridade e envolvimento com a linguagem. (Xiao *et al.*, 2021)

Em termos de ferramentas para desenvolvimento, a *IDE Juno*[21] foi um esforço inicial para criar essa experiência dedicada, no entanto, o desenvolvimento da extensão[22] para o *Visual Studio Code* tomou precedente, contudo, existe também uma solução para *Notebooks* via *Pluto.jl*[23].

[21] Disponível em: https://junolab.org.

[22] Disponível em: https://www.julia-vscode.org.

[23] Disponível em: https://plutojl.org.

4 CONSIDERAÇÕES FINAIS

No geral, Julia traz consigo uma proposta única no mercado de programação, trazendo à tona uma linguagem que procura trazer a praticidade das linguagens de programação de *scripting*, como o *Python* e a performance de linguagens compiladas como *C*, possuindo em seu *design* um nível elevado de integração com o desenvolvimento científico, mitigando problemáticas nesse meio, tais como o problema de se utilizar duas ou mais linguagens para o desenvolvimento de uma única aplicação, bem como, favorece uma modelagem matemática natural dentro do código, vantagens estas que são percebidas através da melhor manutenibilidade de código da aplicação cientifica, proporcionado pela menor ofuscação das ideias que o código contém.

Fora do escopo científico, Julia demonstra seu valor como linguagem de uso geral, ao oferecer uma sintaxe simples de aprendizado, com inferência de tipos ou tipagem explícita, todavia, o uso de Julia de forma menos especializada ainda não é comum, contudo, o esforço por trás de torná-la uma linguagem de fácil adoção é admirável.

Conforme for se intensificando o uso da linguagem Julia e maior for a sua adoção no desenvolvimento de soluções para problemas em diversas áreas da ciência, seu ecossistema passará a alcançar novos patamares, favorecendo o ciclo de produção de algoritmos, de ferramentas e de conhecimento em geral no campo da programação.

REFERÊNCIAS

ARSLAN, Alex *et al*. **Julia Docs**.[S. l.: S. n.], 2023. Disponível em: https://docs.julialang.org/en/v1/. Acesso em: 10 jun. 2023.

BEZANSON, Jeff *et al*. **Julia 1.6 Highlights**. [S. l.], 2021. Disponível em: https://julialang.org/blog/2021/03/julia-1.6-highlights/. Acesso em: 05 jul. 2023.

BEZANSON, Jeff *et al*. Julia: A fresh approach to numerical computing. **SIAM Review**, Filadelfia, v. 59, n. 1, p. 65–98. 2017. Disponível em: https://epubs.siam.org/doi/10.1137/141000671 Acesso em: 10 ago. 2024.

BEZANSON, Jeff *et al*. **Why We Created Julia**. [S. l.], 2012. Disponível em: https://julialang.org/blog/2012/02/why-we-created-julia/. Acesso em: 10 jul. 2023

BOUDREAU, Emma. Julia's Most Awesome Features. **Medium**, [S. l.], 2020. Towards Data Science. Disponível em: https://towardsdatascience.com/julias-most-awesome-features-be51f798f140. Acesso em: 10 jul. 2023.

BUKHARI, Fahren *et al*. Formulation of Sudoku Puzzle Using Binary Integer Linear Programming and Its Implementation in Julia, Python, and Minizinc. **Jambura Journal of Mathematics**, Indonesia, v. 4, n. 2, p. 323-331. 2022. Disponível em: https://ejurnal.ung.ac.id/index.php/jjom/article/view/14194 Acesso em: 10 ago. 2024

CHAGAS, Ronan Arraes Jardim *et al*. Modeling and design of a multidisciplinary simulator of the concept of operations for space mission pre-phase A studies. **Concurrent Engineering**, [S. l.], v. 27. n. 1, p 28–39, 2018. Disponível em: https://doi.org/10.1177/1063293X18804006 Acesso em: 10 ago. 2024.

CHAGAS, Ronan Arraes Jardim. **The SatelliteToolbox.jl for Julia**. [S. l.], 2019.

JULIA COMPUTING. **Case Study**. 2023. Disponível em: https://juliahub.com/case-studies/. Acesso em: 03 ago. 2023.

KILPATRICK, Logan. Julia ranks in the top 5 most loved programming languages for 2022. **Medium**, [S. l.], 2022. JuliaZoid. Disponível em: https://blog.devgenius.io/breaking-julia-ranks-in-the-top-5-most-loved-programming-languages-for-2022-6cb7740240e1. Acesso em: 13 jul. 2023.

KILPATRICK, Logan. Why you should learn Julia, as a beginner / first-time programmer. **Medium**, [S. l.], 2021. Dev Genius. Disponível em: https://blog.devgenius.io/why-you-should-learn-julia-as-a-beginner-first-time-programmer-96e0ad33faba. Acesso em: 13 jul. 2023.

LE, Trang. **10 things I love about Julia**. Filadélfia, 2021. Disponível em: https://trang.page/2021/12/28/10-things-i-love-about-julia/. Acesso em: 15 jul. 2023.

LEITE, Vanessa Matia *et al*. VisuAlg: Estudo de Caso e Análise de Compilador destinado ao ensino de Programação. **Nuevas Ideas en Informática Educativa TISE**, [S. l.], v. 9, p. 637-640, 2013. Disponível em: http://www.tise.cl/volumen9/TISE2013/637-640.pdf. Acesso em: 15 jul. 2023.

ROESCH, Elisabeth *et al*. Julia for biologists. **Nature Methods**, [S. l.], v. 20, p. 655-664, 2023. Disponível em: https://www.nature.com/articles/s41592-023-01832-z Acesso em: 10 ago. 2024

SOBYTE. **Interpreting Julia's 2021: Moving Toward a Mainstream Programming Language**. [S. l.], 2022. Disponível em: https://www.sobyte.net/post/2022-01/julia-2021/. Acesso em: 21 jul. 2023.

STACK OVERFLOW. **Stack Overflow Developer Survey 2022**. [S. l.], 2022. Disponível em: https://survey.stackoverflow.co/2022/. Acesso em: 20 jul. 2023.

STOROPOLI, Jose; HUIJZER, Rik; ALONSO, Lazaro. **Julia Data Science**. [S. l.: s. n.], 2021. Disponível em: https://juliadatascience.io. Acesso em: 13 jul. 2023.

TOMASI, Maurizio; GIORDANO, Mosé. **Towards new solutions for scientific computing: the case of Julia**. [S. n.], [S. l], 2018. Disponível em: https://arxiv.org/pdf/1812.01219 Acesso em: 10 ago. 2024.

VELHO, Roberto Machado *et al*. High Performance Computing in Julia. In: LORENZON, Arthur; CASTRO, Márcio; PILLON, Mauricio. **Minicursos da XXII Escola Regional de Alto Desempenho da Região Sul**. Porto Alegre: Sociedade Brasileira de Computação, 2022. p. 26-63. Disponível em: https://books-sol.sbc.org.br/index.php/sbc/catalog/view/83/366/630 Acesso em: 10 ago. 2024

XIAO, Lei *et al*. Julia Language in Computational Mechanics: A New Competitor. **Archives of Computational Methods in Engineering**, [S. l.], v. 29, p. 1713-1726, 2021. Disponível em: https://link.springer.com/article/10.1007/s11831-021-09636-0 Acesso em: 10 ago. 2024