Biblioteca para exemplificação no ensino de Álgebra Linear

Bruno Hryniewicz
brunohry@ic.ufrj.br
Instituto de Computação, UFRJ

Fernando França
fernandofranca@ic.ufrj.br
Instituto de Computação, UFRJ

João Paixão
jpaixao@ic.ufrj.br
Instituto de Computação, UFRJ

Laura O. Moraes
laura@uniriotec.br
Programa de Pós-Graduação em Informática, UNIRIO

RESUMO

Existe uma grande interseção entre a matemática e a computação que muitas vezes não é perceptível ao aluno do curso de computação. Uma abordagem utilizada para tornar essa relação mais visível e aumentar o engajamento dos alunos com as disciplinas matemáticas é a exemplificação. Neste artigo, apresentamos uma biblioteca em Julia construída para facilitar a exemplificação em situações práticas dos conceitos de álgebra linear. A biblioteca atua como uma camada de abstração, encapsulando objetos matriciais mais complexos, como imagens e sinais. Como resultado, apresentamos implementações das interações desses objetos com os conceitos da álgebra linear, abstraindo esse trabalho dos usuários, em sua maioria, iniciantes. Dessa maneira, é possível com poucas linhas de código demonstrar a aplicabilidade da teoria vista em sala de aula, transformando um exemplo feito com vetores e pontos (teórico) em um exemplo aplicado a imagens e sinais (prático).

CCS CONCEPTS

• **Social and professional topics** → *Computing education*; • **Applied computing → Interactive learning environments**.

PALAVRAS-CHAVE

educação de computação, álgebra linear, recurso educacional, biblioteca em Júlia, ferramenta

1 INTRODUÇÃO

Existe uma grande interseção entre matemática e computação, dado que o nascimento de computadores foi motivado pela necessidade de resolver problemas matemáticos complexos. Com o avanço da ciência da computação, tais interseções acabam sendo ocultadas por camadas sobrepostas de abstração, camuflando o conceito matemático de resolver problemas contidos no ato de criar *softwares* [5]. Esses níveis de abstrações são essenciais para o avanço tecnológico, visto que facilitam a interação homem-máquina. Um exemplo disso é como a adição de uma interface gráfica em sistemas computacionais conseguiu facilitar sua utilização criando uma abstração em cima do que antes era linha de comando [26]. Com este aparente distanciamento entre a computação contemporânea e sua origem matemática, não fica claro para os novos estudantes da área de computação a necessidade de tantas cadeiras de matemática em seu curso [3]. Sob este pretexto, existem alunos que não se motivam a estudar tais temas durante suas graduações e podem com isso ter seus desempenhos acadêmico e profissional prejudicados [3].

Torna-se um desafio ao corpo acadêmico diminuir o distanciamento entre matemática e computação. Para demonstrar a importância da álgebra linear na área de computação, principalmente no contexto atual de popularização de aplicações que utilizam inteligência artificial, apresentamos três exemplos de aplicações:

- Aplicativos de "GPS" que fornecem cálculo de melhor rota, onde se utiliza da resolução de problemas de caminhos mínimos e otimização [4] para recomendar o trajeto mais rápido.
- A computação gráfica, é uma área de pesquisa que tem como fundamento a álgebra linear. Alguns produtos construídos com base na computação gráfica são jogos eletrônicos, filmes, simuladores, imagens médicas e visualizações científicas [24].
- Aplicações de aprendizado de máquina utilizam álgebra linear como ferramenta de sua construção [1], então aplicações como processamento de linguagem natural, detecção de fraudes e tantos outros tópicos têm suas bases na álgebra linear.

No entanto, nas disciplinas iniciais nos cursos de graduação em computação, os alunos ainda não conectam a aplicabilidade com as disciplinas teóricas, obtendo a percepção de um desalinhamento do curso com o mercado de trabalho [20]. Um segundo desafio para os professores das cadeiras matemáticas em cursos de graduação em computação é a dificuldade de aprendizado dos alunos nestas disciplinas [14]. Historicamente, os jovens do Brasil apresentam déficits na competência matemática quando comparado à média da Organização para a Cooperação e Desenvolvimento Econômico [9], sendo este um dos motivadores que cria nos professores a percepção da dificuldade no ensino de matemática para o ensino superior [8, 14]. Aliado a este déficit, há uma falsa impressão da não necessidade da matemática na formação da computação. Alunos entram no curso atraídos pelas ofertas do mercado de trabalho [11] e mantêm seu foco em entender quase exclusivamente programação. Assim, acabam realizando as cadeiras matemáticas apenas por obrigação, pois não percebem ou não dão a devida importância às suas aplicações na computação. Estas cadeiras matemáticas se tornam grande parte da dificuldade do curso, fato este, evidenciado pelos grandes índices de reprovação das cadeiras de cálculo diferencial e integral da Universidade Federal do Rio de Janeiro (UFRJ) [15].

EduComp'24, Abril 22-27, 2024, São Paulo, São Paulo, Brasil (On-line)

Uma abordagem que pode ser utilizada para aumentar o engajamento e interesse dos alunos de computação nas disciplinas matemáticas é a exemplificação [2], trazendo significado aos conceitos aprendidos pelos estudantes [12]. Neste artigo apresentamos o recurso educacional EasyLinalg, uma biblioteca em Julia[1] construída para facilitar a exemplificação dos conceitos abstratos comumente utilizados em álgebra linear, focando em avançar nas seguintes questões:

Q1 Como agilizar a construção de exemplos práticos e complexos de álgebra linear em sala de aula?

Q2 Como prover autonomia a alunos iniciantes na construção de exemplos práticos e complexos de álgebra linear?

2 FUNDAMENTAÇÃO TEÓRICA

Na década de 90, havia uma preocupação de que o currículo ensinado de álgebra linear nas universidades americanas não estava adequado às demandas dos diferentes campos do conhecimento que utilizam esta disciplina na solução de problemas, como engenharias, ciência da computação, estatística, entre outros [6]. Então, em 1993, o Linear Algebra Curriculum Study Group se reuniu e publicou um conjunto de recomendações para o ensino de álgebra linear em um curso introdutório. Entre as recomendações se encontra: valorizar a importância das aplicações da álgebra linear aos alunos, o uso de tecnologia para apoiar o curso e o curso ser orientado a matrizes [6]. Em 2022, essas recomendações foram revisadas [27], destacando ainda mais a importância desta disciplina em áreas da computação como ciência de dados, processamento de sinais, criptografia, entre outras, e reforçando a utilização de tecnologia no seu ensino.

A partir das recomendações iniciais, o projeto ATLAST (ainda na década de 90) desenvolveu e publicou um conjunto de projetos de computador adequados para aulas de álgebra linear. Tanto nesse projeto como na versão atualizada das recomendações, é sugerido que os alunos utilizem programas com extensas rotinas simbólicas, geométricas e exatas como o GeoGebra[2], MATLAB[3], MAPLE[4] e SAGE[5] para a resolução desses problemas. A utilização desses programas permite que alunos criem seus próprios exemplos, explorem e descubram os teoremas e consigam verificar resultados de forma rápida e correta [13], interagindo e investigando os objetos de estudo [17]. Ainda, a utilização dessas ferramentas para resolução de problemas de baixas dimensões (com fácil visualização) pode ajudar na compreensão dos conceitos pelos alunos. As recomendações atualizadas ainda incentivam a utilização de aplicações como uma forma de aumentar o interesse dos alunos pela disciplina, motivar o ensino dos conceitos e contextualizar os problemas através de exemplos práticos [27].

Uma pesquisa [19] realizada com 28 alunos (da "Geração Z") de um curso de engenharia, onde o curso utilizou o aplicativo GeoGebra como ferramenta didática, indicou que esses alunos acreditam que têm mais facilidade em aprender fazendo em comparação com outras modalidades como assistindo, ouvindo, escrevendo ou lendo. Ainda, uma combinação de aula que envolva a voz, o quadro e softwares/aplicativos obteve 53.6% da preferência entre os alunos. Essa pesquisa aponta que incorporar recursos didáticos interativos nas aulas matemáticas são um caminho para mitigar a dificuldade de aprendizado dos alunos em tais cursos.

[1]https://julialang.org/
[2]https://www.geogebra.org/
[3]https://www.mathworks.com/
[4]https://www.maplesoft.com/products/maple/
[5]https://www.sagemath.org/

3 TRABALHOS RELACIONADOS

Macêdo *et al.* [16] descrevem softwares livres para auxiliar no ensino de matemática. Entre os recomendados para álgebra linear, se encontram o Winmat (desativado), GeoGebra e Winplot[6]. O Winmat e o Winplot são aplicativos disponíveis para a plataforma Windows enquanto o Geogebra está disponível em inúmeras plataformas, inclusive online. O Winmat fazia somente cálculos matriciais. O Winplot pode ser utilizado na resolução visual de sistemas lineares [16]. Dentre as ferramentas encontradas, o GeoGebra é o recurso educacional mais utilizado, permitindo a visualização e manipulação de vetores e pontos em espaços bi e tridimensionais. Sua versatilidade permite sua utilização em aulas de álgebra linear para resolução de problemas no contexto de engenharias [12, 17, 19], física, meteorologia [17], e oficinas para capacitação de alunos do bacharelado e licenciatura matemática e professores da Educação Básica e Ensino Superior [23].

Complementando a pesquisa, foi buscado no Portal de Periódicos da CAPES e no Google Scholar pelos termos "ferramenta álgebra linear", "software álgebra linear" e "biblioteca álgebra linear" em outubro de 2023. Foram encontradas ainda três ferramentas. A primeira é o LineAlg [25], um aplicativo de celular para estudo de álgebra linear no ensino médio, onde é possível fazer cálculos com matrizes como inversa, determinante e transposta, além de solucionar um sistema linear. A segunda é o AlgLin [18], um software web (desativado) que propõe exercícios e faz operações matriciais. Por fim, foi encontrada a ferramenta LinA [7], um aplicativo para celulares Android (desativado) que simula uma calculadora matricial passo a passo, fazendo operações de escalonamento, determinantes e inversão de matrizes. A Tabela 1 apresenta uma comparação entre as ferramentas apresentadas e a biblioteca proposta.

Os exemplos comumente utilizados em sala de aula, como os construídos com o GeoGebra, são algébricos com sua complementação geométrica. Apesar de serem importantes para construção da intuição do conceito, esses exemplos não contribuem para o entendimento da aplicação da álgebra linear. Linguagens de programação como MATLAB, Python, Julia entre outras, permitem criar exemplos mais práticos e complexos. No entanto, é preciso entender, manipular e transformar os diferentes tipos de entrada para conseguir realizar as operações aprendidas em sala de aula.

Portanto, no contexto atual de aplicações que utilizam grandes volumes de dados n-dimensionais e tipos de dados diversos, principalmente no campo de inteligência artificial, se tornou necessário um recurso que permita ao usuário a manipulação e visualização de tipos de dados complexos e contextualizados em suas aplicações. A exemplo de outras ferramentas, como o Scikit-learn [28], que traz um conjunto de componentes de aprendizado de máquina em Python onde algoritmos de alto grau de complexidade estão abstraídos por trás de chamadas de funções, este artigo apresenta a biblioteca EasyLinalg. Esta é uma biblioteca criada em Julia que utiliza uma abordagem de abstração dos tipos de entrada para demonstrar como um software pode lidar com a generalização das

[6]https://winplot.softonic.com.br/

Tabela 1: Comparação entre os *softwares* que auxiliam o ensino de álgebra linear

Software	Operações	Disponibilização D = desativado	Tipos de entrada	Visualização de gráficos
Winmat [16]	Cálculos matriciais	Software desktop (D)	Matriz	Não
Winplot [16]	Resolução visual de sistemas lineares	Software desktop	Equações	Sim
GeoGebra [16]	Visualização e manipulação de vetores e pontos	Software desktop; Software web	Ponto e vetor	Sim
LineAlg [25]	Cálculos matriciais	Aplicativo de celular	Matriz; Equações	Não
AlgLin [18]	Cálculos matriciais	Software web (D)	Matriz	Não
LinA [7]	Cálculos matriciais	Aplicativo de celular (D)	Matriz	Não
EasyLinalg	Cálculos matriciais; visualização e manipulação de diferentes tipos de dados; resolução de sistemas lineares; combinação convexa; iteração de Cadeia de Markov	Biblioteca para linguagem Julia	Matriz, Ponto, Vetor, Imagem e Sinal	Sim

aplicações de álgebra linear de uma maneira alto nível. Ela se difere das ferramentas anteriores por abstrair tipos de dados mais complexos, além de vetores e matrizes, para realizar operações aprendidas no curso de álgebra linear, aproximando a aplicação da teoria e visando proporcionar um aprendizado significativo aos alunos.

4 METODOLOGIA PARA DESENVOLVIMENTO DA BIBLIOTECA

Esta seção apresenta cada etapa da metodologia ágil utilizada para o projeto e desenvolvimento da biblioteca. No desenvolvimento ágil, a ideia principal é fornecer protótipos funcionais para serem testados pelos usuários em versões iterativas de quatro etapas [22]: planejamento, projeto, implementação e validação.

4.1 Planejamento

A etapa do planejamento consiste em definir o público-alvo e os objetivos que se deseja alcançar com a construção da ferramenta. A EasyLinalg é uma biblioteca cujo objetivo é facilitar a demonstração de exemplos de álgebra linear, buscando aproximar a teoria da prática em uma tentativa de melhorar a absorção de conhecimento dos estudantes. Atua criando uma camada de abstração, reduzindo os conhecimentos necessários de programação para realizar exemplificações de conteúdos da matéria. Portanto, a biblioteca deve:

- Facilitar demonstrações de aplicações da álgebra linear utilizando tipos de dados matriciais que tenham utilizações conhecida para os alunos, como imagens, sinais e vetores.
- Abstrair todo o trabalho de carregar e imprimir essas categorias de dados no computador.

Com isso, objetiva-se que uma pessoa com pouco conhecimento de programação possa explorar essas estruturas da álgebra linear de maneira visual e próxima a aplicações reais. Ainda, deve ser possível promover a autonomia do aluno para testar e experimentar novas possibilidades independentemente dos exemplos dados em sala de aula, ficando a cargo de sua imaginação usar novos parâmetros e valores nos exercícios.

4.1.1 Usuários. Entender o público alvo é um ponto crucial para criar uma biblioteca que tenha potencial de utilização. Para isso, foram definidas três proto-personas [10] de utilização do projeto: o Professor, o Aluno e a Desenvolvedora.

(1) **Proto-persona do Professor** A proto-persona do Professor é caracterizada como aquele que está ensinando álgebra linear. Seu conhecimento de programação é, ao menos, básico e seu conhecimento de álgebra linear é avançado. Seu caso de uso na EasyLinalg é desenvolver exemplos de maneira rápida, onde facilmente ele possa alterar as características de seu exemplo, trazendo dinamicidade para aula. A EasyLinalg para o professor deve ser um otimizador de tarefas e um simplificador de exemplos, tornando-os mais palpáveis e atraentes.
(2) **Proto-persona do Aluno** O Aluno é aquele que está aprendendo. Seu conhecimento de álgebra linear ainda é pequeno, seu conhecimento de programação é básico e nunca teve contato com a linguagem Julia. Para o Aluno, seu maior ganho é a simplicidade que a EasyLinalg trará, fazendo com que os exemplos dados pelo professor sejam mais simples de serem entendidos. Essa simplicidade também pode ser convertida em autonomia, assim facilitando o aluno a explorar sozinho os tópicos da álgebra linear. Por último, os exemplos dados para os alunos podem ter mais conexão com aplicações reais, elevando o seu nível de interesse.
(3) **Proto-persona da Programadora** A Programadora se caracteriza por uma pessoa com conhecimento pelo menos mediano em programação e álgebra linear e com interesse em evoluir a biblioteca. A Programadora é ou já foi uma das personas anteriores, e agora ajudará na evolução do projeto. Para a Programadora, seu caso de uso é ajudar a melhorar a experiência do aluno e do professor, buscando fazer com que a EasyLinalg se desenvolva e seja mais utilizada.

4.1.2 Exemplo de aplicação. Para demonstrar as abstrações que a biblioteca deveria implementar, definiu-se que a EasyLinalg deveria oferecer uma exemplificação próxima à demonstração a seguir:

- **Exemplo algébrico de reflexão de um ponto** Ao explicar uma transformação linear de reflexão, por exemplo, podemos dizer que a matriz $\left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$ espelha um ponto do $\mathbb{R}^2$ ao redor do eixo X, ou seja, se multiplicarmos um ponto do $\mathbb{R}^2$ por essa matriz, ele terá sua coordenada Y invertida:

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -2 \end{bmatrix}$$

- **Exemplo geométrico de reflexão de um ponto** Após o exemplo numérico da reflexão, podemos enriquecê-lo com um gráfico 2D do que aconteceu. As Figuras 1 e 2 apresentam a representação geométrica em $\mathbb{R}^2$ do ponto e do ponto resultante após a reflexão.

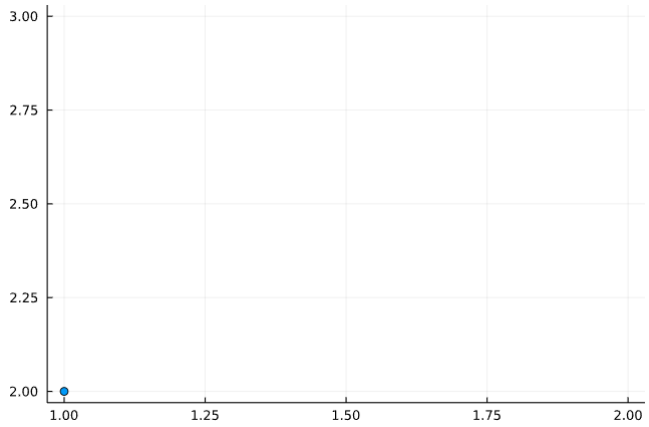

Figura 1: Ponto (1,2) no $\mathbb{R}^2$

- **Exemplo prático de utilização de reflexão** Por fim, entendida as partes algébrica e geométrica, é possível realizar uma exemplificação de utilização deste conceito. Para isso utilizaremos a imagem da Figura 3. Uma imagem é computacionalmente representada por uma matriz de píxeis, sendo um píxel um ponto no $\mathbb{R}^2$ com uma intensidade de cor associada a ele. Pode-se então aplicar em cada píxel da imagem a reflexão vista anteriormente e o resultado será a reflexão da imagem toda, considerando o centro da foto como origem de um plano cartesiano. Seguindo com o exemplo, aplicaremos em cada píxel da imagem a matriz de reflexão, obtendo o resultado da Figura 4. Entretanto, existe uma particularidade quando se trata de imagens. Quando falamos geometricamente do ponto [1, 2], é entendido que a coordenada X deste ponto vale 1 e a coordenada Y vale 2. Já no âmbito de computação gráfica, a representação dos eixos X e Y do plano cartesiano convencional é invertida. Quando falamos do píxel de posição [1, 2], estamos falando de um píxel de coordenada X com valor 2 e coordenada Y de valor 1. Esse é o padrão da representação de imagens em computadores, e dado este padrão, ao aplicar a mesma matriz anterior nos píxeis da imagem, teremos uma reflexão ao redor do eixo Y e não no eixo X.

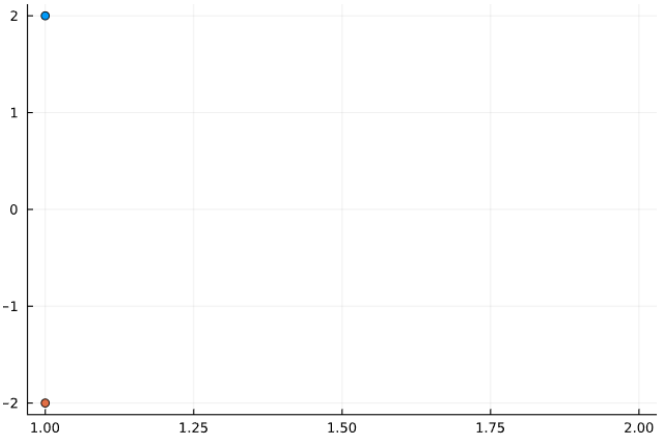

Figura 2: Pontos (1,2) e (1,-2) no $\mathbb{R}^2$

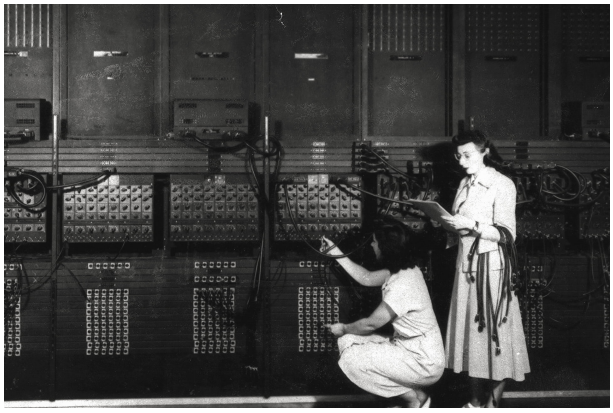

Figura 3: Figura original

4.2 Projeto

O projeto define como será a interação do usuário com o sistema e as tecnologias utilizadas para seu desenvolvimento.

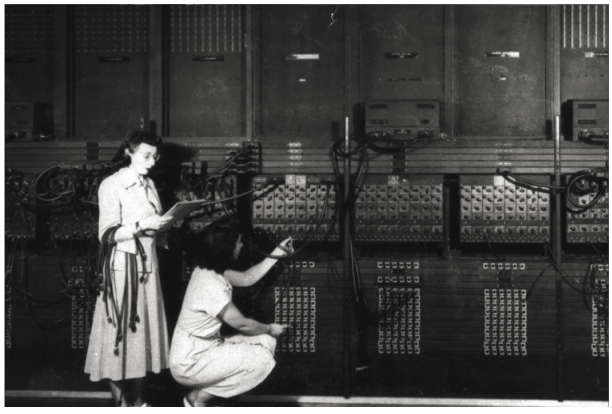

Figura 4: Figura após reflexão

4.2.1 Interação com o usuário. Durante o estudo da álgebra linear, em seu processo de exemplificação, são geralmente utilizados objetos como pontos no espaço, vetores e matrizes. Porém, computacionalmente existem muitos outros objetos com características matriciais que podem ser muito mais ilustrativos para o ensino. A maneira de realizar a entrada e depois exibir esses objetos em linguagens de programação é uma barreira à utilização de exemplos como imagens e sinais. Se pudermos encapsular estes objetos de maneira a manter sua semântica e característica, e dentro deste encapsulamento já estiver pré-estabelecido como fazer a entrada de dados, como realizar operações com esses dados e como mostrar essas informações, será muito reduzido o nível de complexidade necessária para realizar a entrada e saída de um dado, além de agregar informações semânticas para facilitar a interpretação dos resultados. Essa redução de complexidade pode então ser extrapolada para se trabalhar com objetos matriciais mais complexos sem aumentar a complexidade para quem utiliza. Foram então definidos os tipos de dados que seriam úteis para exemplificação dos conceitos. O diagrama da Figura 5 mostra a interação planejada do usuário com a biblioteca. O usuário define o tipo de entrada e a operação a ser realizada. Internamente, a biblioteca traduz essas entradas, criando os objetos com os tipos correspondentes, realizando conversões entre tipos complexos (como vetores, imagens e sinais) e tipos simples (como uma matriz) e realizando as operações necessárias. A biblioteca atua como uma camada de abstração da criação e conversão entre esses tipos. A interpretação do objeto ocorre na entrada do dado, então o objeto é carregado no tipo de dado específico do seu domínio. Um exemplo prático é o caso de uma imagem. Para carregar essa imagem dentro da EasyLinalg, deve-se utilizar o tipo de dado "*Image*", e então todo o comportamento de imagem implementado pela biblioteca já estará disponível para esse objeto. A Tabela 2 apresenta os tipos implementados e o comando necessário para a construção de seus objetos.

4.2.2 Operações. Para o escopo inicial da EasyLinalg, foram previstas as seguintes operações: soma (+) e subtração (-) entre objetos do mesmo tipo, multiplicação de um tipo por um escalar ou por uma matriz numérica (*), divisão de um tipo por um escalar (/) e divisão pela esquerda (\) de uma matriz numérica por um tipo, representando a resolução de um sistema linear. A resolução do sistema linear é a operação inversa da multiplicação por uma matriz numérica. Operações mais complexas como a combinação convexa e iteração da cadeia de Markov também foram adicionadas. A combinação convexa consiste na transformação gradual de um objeto em outro. Já a operação em cadeia de Markov aplica uma matriz várias vezes a um tipo, normalmente utilizado para representar transições de estados em sistemas lineares dinâmicos. As operações mencionadas devem ocorrer entre dois objetos de tipos iguais ou entre um objeto de tipo da EasyLinalg e uma matriz numérica ou vetor. Um resumo dessas operações pode ser visto na Tabela 3.

4.3 Implementação

O projeto foi construído utilizando Julia como linguagem de programação por esta possuir sintaxe próxima à notação matemática, o que se adéqua ao objetivo de simplificar demonstrações de álgebra linear. Esse não é o único aspecto dessa decisão, pois seguindo a mesma linha de raciocínio, o MATLAB apresenta a mesma abordagem e, segundo o Stack Overflow Survey 2021 [21], ele possui uma popularidade cerca de 3 vezes maior do que Julia. O segundo grande fator para a escolha da linguagem vem do fato de Julia ser código aberto, sendo esse um fator que democratiza o acesso à ferramenta e permite uma maior colaboração entre indivíduos interessados em desenvolver o projeto, tendo então um grande peso para a decisão da escolha da linguagem.

4.4 Validação

A validação da biblioteca ocorreu através da criação de um roteiro de aula em que se utiliza os tipos e operações descritas, alternando entre a notação algébrica, geométrica (tipo teórico) e um tipo prático. A validação buscou responder às perguntas de pesquisa verificando a quantidade de linhas de código necessária para criar um exemplo novo, transformar um exemplo teórico (com vetores e pontos) em um exemplo prático (imagens e sinais) — investigando a Q1 — e a facilidade de leitura e entendimento dos códigos apresentados, relacionada a Q2. Esses resultados serão detalhados na seção a seguir.

5 RESULTADOS

Nesta seção será refeita a demonstração da exemplificação desejada descrita na seção 4.1.2 utilizando a EasyLinalg, com intuito de mostrar a simplicidade da utilização da biblioteca. Outros exemplos utilizando diferentes tipos e aplicações também serão apresentados. Um ponto a se observar é como um par de exemplos teórico e prático utilizam sempre o mesmo código, mudando apenas os tipos de entrada.

5.1 Transformação linear de um objeto

Uma transformação linear é uma operação que move um objeto entre dois espaços vetoriais. Sua aplicação é muito importante no âmbito da computação gráfica, onde é muito utilizada para manipular imagens. Basicamente uma transformação linear é dada por $A \times B = C$, onde A é o objeto de origem, B é a matriz que transforma esse objeto e C o resultado. A reflexão apresentada na seção 4.1.2 é um exemplo de transformação linear.

Ao explicar uma transformação linear de reflexão, podemos dizer que a matriz A definida no código abaixo a esquerda espelha um ponto do $\mathbb{R}^2$ ao redor do eixo X, ou seja, se for multiplicado um ponto do $\mathbb{R}^2$ por essa matriz, ele terá sua coordenada Y invertida. O código ilustra como essa operação seria feita utilizando a biblioteca proposta, resultando nos gráficos das Figuras. 1 e 2. Agora, para melhor exemplificar, podemos ilustrar onde isso pode ser aplicado na computação. Para isso o código abaixo a direita pode ser utilizado para exemplificar utilizando imagens ao invés de pontos. Repare que o código se mantém o mesmo, exceto pelo tipo utilizado na criação do objeto. Esse código gera as imagens das Figuras 3 e 4.

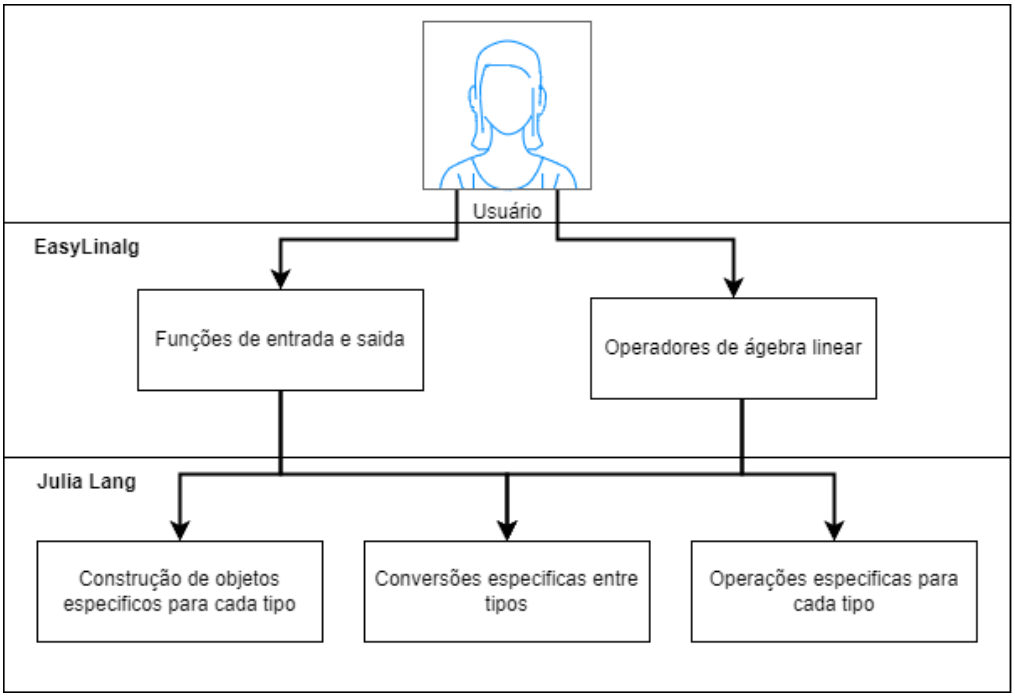

Figura 5: Interação do Usuário com a EasyLinalg.

Tabela 2: Tipos atualmente desenvolvidos na EasyLinalg, com sua descrição e exemplo de construção em código

Categoria	**Tipo do dado de entrada**	**Descrição**	**Construção**
Teórico	Seta 2D	Setas, de duas ou três dimensões, são representações vetoriais do $\mathbb{R}^2$ ou $\mathbb{R}^3$ respectivamente, centradas na origem. Suas construções recebem como parâmetro os valores das respectivas coordenadas.	Arrow2D(x, y)
	Seta 3D		Arrow3D(x, y, z)
	Ponto 2D	Pontos, de duas ou três dimensões, são representações de pontos do $\mathbb{R}^2$ ou $\mathbb{R}^3$ respectivamente. Suas construções recebem como parâmetro os valores das respectivas coordenadas.	Point2D(x, y)
	Ponto 3D		Point3D(x, y, z)
Prático	Sinal	Sinais são representações de duas dimensões de funções. Podem ser construídos a partir de uma função e um intervalo numérico onde essa função será avaliada, ou através 2 objetos vetoriais, onde um contém as coordenadas do eixo X e o outro, as coordenadas do eixo Y do sinal.	SignalBuildU(Função, Inicio:Incremento:Fim)
			SignalBuild([X1, X2,...,Xi], [Y1, Y2,...,Yi])
	Imagem	Imagens são construídas a partir de arquivos de imagens.	Image("caminho/do/arquivo.formato")

Tabela 3: Operações implementadas na EasyLinalg

Operação	Descrição	Exemplo de declaração
+	Adição	Point2D(1,2) + Point2D(4,9)
-	Subtração	Point3D(2,2,3) - Point3D(1,2,5)
*	Mulplicação por escalar	Arrow2D(1,2) * 4
*	Multiplicação por matriz	Arrow2D(1,2) * [1 2; 3 4]
/	Divisão entre tipo e escalar	Point2D(8,7) / 3
\	Resolução de sistemas	[Arrow2D(2,2); Arrow2D(6,4)] \Arrow2D(4,9)
convex_combination	Combinação convexa	convex_combination(Array, step)
RunMarkovChain	Iteração de cadeia de Markov	RunMarkovChain(signal, A, n)

Código 1: Transformação linear de um ponto 2D

```
using EasyLinalg
A = [1 0;
     0 -1]

B = Point2D(1,2)

C = B * A

Draw(B)
Draw(C, true)
```

Código 2: Transformação linear aplicada a uma imagem

```
using EasyLinalg
A = [1 0;
     0 -1]

B = Image("eniac.jpg")

C = B * A

Draw(B)
Draw(C)
```

5.2 Multiplicação por escalar e soma

A fim de demonstrar a utilização das operações de multiplicação por escalar e soma entre tipos, utilizaremos o exemplo de média aritmética entre dois objetos de mesmo tipo da EasyLinalg. Neste exemplo, utilizaremos os objetos que representam um vetor em 2D e um sinal. Os códigos abaixo produzem as Figuras 6 e 7. Essas operações podem ser demonstradas ao aluno de forma clássica, utilizando vetores e depois exemplificadas em uma aplicação com sinais (por exemplo, uma mixagem de sons) utilizando o mesmo código e trocando o objeto utilizado por um sinal.

Código 3: Multiplicação por escalar e soma de dois vetores 2D

```
using EasyLinalg
A1 = Arrow2D(2,2)

A2 = Arrow2D(6,4)

Draw([A1, A2])

B = 0.5 * A1 + 0.5 * A2
Draw(B, false)
```

Código 4: Multiplicação por escalar e soma de dois sinais

```
using EasyLinalg
A1 = SignalBuildU(
    x -> cos(100*x),
    -10:0.1:10)

A2 = SignalBuildU(
    x -> 10*sin(x),
    -10:0.1:10)

Draw([A1, A2])

B = 0.5 * A1 + 0.5 * A2
Draw(B, false)
```

5.3 Resolução de um sistema linear

A resolução de sistemas lineares pode ser utilizada para decompor objetos como os das Figuras 6 e 7, onde $B = 0.5 * A1 + 0.5 * A2$. Para exemplificar com os objetos da EasyLinalg, utilizaremos o seguinte produto vetorial

```
A x P = B
```

Onde A é um vetor de tipos da EasyLinalg e P um vetor coluna de tipos numéricos (pesos escalares). O resultado B é uma soma dos itens de A ponderados por P, mantendo o tipo dos itens de A. Podemos então aplicar o operador contra-barra (\) para resolver o problema de achar P tendo A e B, e descobrindo a porção que cada elemento de A contribui em B. Note que este exemplo é a operação contrária a operação do exemplo anterior (Seção 5.2). Continuando o código anterior, se fizermos a operação

```
A = [A1, A2]
C = A\B
```

teremos como resultado o vetor

$$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

representando o valor da ponderação de cada vetor ($A1$ e $A2$) na construção de B. Observe que o código pode ser utilizado em ambos os exemplos apresentados na Seção 5.2 (ponto 2D e sinal), sendo independente do tipo de entrada da EasyLinalg.

5.4 Combinação convexa: *morphing*

O termo *morphing* consiste na transição de um objeto em outro. Ele é muito utilizado para produzir efeitos especiais em filmes, como uma transição entre cenas. Esta transformação começa com 100% do primeiro objeto e, a cada iteração, diminui uma pequena

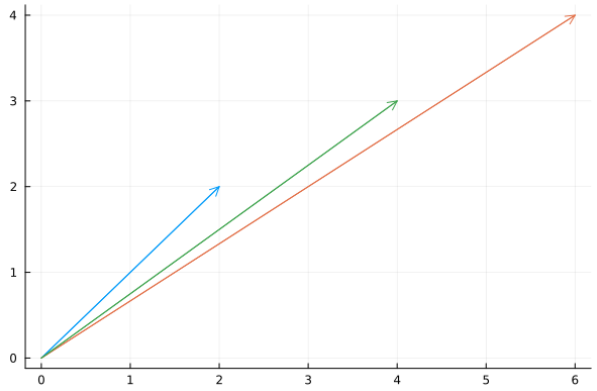

Figura 6: Vetores originais em azul e laranja e a resultante ponderada em verde

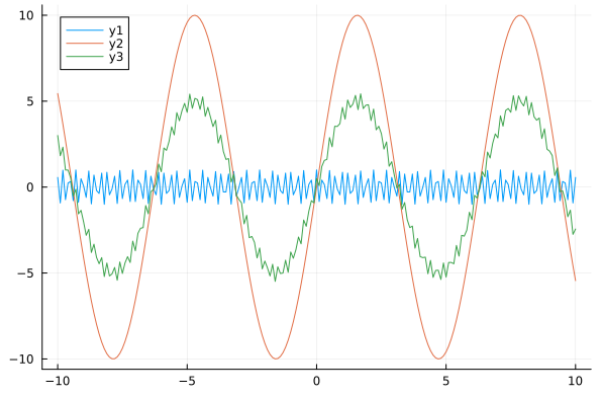

Figura 7: Sinais originais em azul e laranja e a resultante ponderada em verde

porcentagem do primeiro objeto e adiciona a mesma proporção do segundo objeto, terminando então com 100% do segundo objeto. No exemplo a seguir, ilustraremos esta aplicação em um ponto 3D e uma imagem. A Figura 8 ilustra em uma mesma figura o ponto se afastando do início e se aproximando do fim em incrementos de 25%. Já nas Figuras 9- 13, a imagem resultante "se afasta" da imagem original e "se aproxima" da imagem final em incrementos de 25% a cada passo.

Código 5: Combinação convexa entre dois pontos 3D

```
using EasyLinalg
inicio = Point3D(1,1,1)
fim = Point3D(10,10,10)

Draw([inicio, fim])

B = convex_combination(
        [inicio, fim], 0.25)
Draw(B)
```

Código 6: Combinação convexa entre duas imagens

```
using EasyLinalg
inicio = Image("ally.jpg")
fim = Image("lion.jpg")

Draw([inicio, fim])

B = convex_combination(
        [inicio, fim], 0.25)
Draw(B)
```

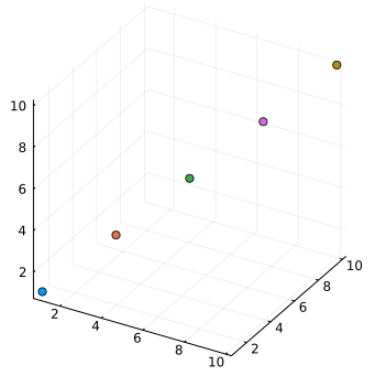

Figura 8: Ponto 3D se afastando do ponto inicial (1,1,1) e se aproximando do ponto final (10,10,10) em incrementos de 25%

Outras aplicações como diferentes transformações lineares e iteração de cadeia de Markov podem ser vistas na documentação da biblioteca[7] ou no Google Colab de demonstração[8].

6 CONSIDERAÇÕES FINAIS

Este trabalho visou diminuir a distância entre a matemática e a computação em sala de aula focando em exemplificações de álgebra linear através do desenvolvimento de uma biblioteca que possibilita a construção de exemplos teóricos e práticos. Foram disponibilizados seis tipos de objetos e oito operações, que podem ser aplicadas a qualquer um dos tipos. Após o devido planejamento e implementação, foi possível exemplificar os conceitos utilizando poucas linhas de código e navegar entre exemplos teóricos e práticos utilizando o mesmo código, trocando apenas a linha de código que contém o tipo do objeto apresentado. Essas duas características agilizam a construção de diferentes exemplos para um mesmo conceito apresentado, respondendo à questão de pesquisa 1. O conceito-chave do trabalho é a tipificação dos diferentes objetos que podem ser representados por uma matriz. Através desses tipos é possível implementar a interação daquele objeto com os conceitos da álgebra linear, abstraindo esse trabalho dos usuários e provendo autonomia aos usuários para a construção de exemplos personalizados. Além disso, foi escolhida a linguagem Julia para a criação da biblioteca, que possui uma sintaxe parecida a utilizada em aulas de álgebra

[7] https://github.com/MachineTeachingEdu/Easylinalg
[8] https://colab.research.google.com/drive/1RpRDbjQ_qcIKANMzBbbKkblaeupvpL9-

Figura 9: 100% de A1 e 0% de A2 **Figura 10: 75% de A1 e 25% de A2** **Figura 11: 50% de A1 e 50% de A2** **Figura 12: 25% de A1 e 75% de A2** **Figura 13: 0% de A1 e 100% de A2**

linear e nomes significativos para os tipos dos objetos, tornando a leitura do código fácil e fluida, respondendo à questão 2. A escolha da linguagem e organização do projeto permite que novos tipos possam ser criados e incorporados à biblioteca, para conseguir gerar um portfólio cada vez mais amplo de exemplos. Em resumo, a vantagem em se utilizar a biblioteca EasyLinalg em comparação com os outros *softwares* apresentados é a transposição entre exemplos teóricos e práticos de forma rápida, e que não sejam exemplos estáticos.

6.1 Limitações

A ferramenta apresentada por ser uma biblioteca para uma linguagem de programação, possui uma flexibilidade na construção dos exemplos, permitindo aos alunos interagir e explorar. No entanto, essa liberdade possui o custo do usuário ainda necessitar de conhecimentos básicos de programação por parte dos professores e alunos que a utilizarão. Tendo como público-alvo professores e alunos da graduação em computação, tentamos mitigar essa limitação disponibilizando um roteiro de aula já programado no Google Colab de demonstração. A partir do roteiro, os alunos e professores podem modificá-lo para investigar as questões durante a aula, diminuindo a sobrecarga de construção de exemplos desde o início.

6.2 Trabalhos Futuros

Os direcionamentos futuros desta pesquisa envolvem o teste e a utilização do projeto em instituições de ensino, não se prendendo somente a salas de aulas físicas, mas também em cursos online, mentorias ou *bootcamps*. Planeja-se a realização de um workshop para apresentação e demonstração da ferramenta, onde os usuários terão a oportunidade de testá-la. Neste workshop será conduzida uma pesquisa de opinião de usabilidade sobre sua facilidade de uso e utilidade em sala de aula. Ainda, os resultados dessa pesquisa direcionarão os novos tipos a serem incorporados na biblioteca (por exemplo, grafos) e melhorias nas funcionalidades atuais.

REFERÊNCIAS

[1] Charu C. Aggarwal. 2020. *Linear Algebra and Optimization for Machine Learning* (1 ed.). Springer. https://doi.org/10.1007/978-3-030-40344-7

[2] Cahit Aytekin and Yasemin Kiymaz. 2019. Teaching Linear Algebra Supported by GeoGebra Visualization Environment. *Acta Didactica Napocensia* 12, 2, 75–96.

[3] Theresa Beaubouef. 2002. Why computer science students need math. *ACM SIGCSE Bulletin* 34, 4, 57–59.

[4] G.C. Calafiore and L. El Ghaoui. 2014. *Optimization Models*. Cambridge University Press.

[5] Martin Campbell-Kelly. 2009. Origin of computing. *Scientific American* 301, 3, 62–69. https://doi.org/10.1038/scientificamerican0909-62

[6] David Carlson, Charles R. Johnson, David C. Lay, and A. Duane Porter. 1993. The Linear Algebra Curriculum Study Group Recommendations for the First Course in Linear Algebra. *The College Mathematics Journal* 24, 1, 41–46. https://doi.org/10.1080/07468342.1993.11973504

[7] Cleriston Dantas and Vaston Gonçalves da Costa. 2016. LinA como ferramenta auxiliar no processo de ensino-aprendizagem de Álgebra Linear. In *Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE 2016)*, Vol. 5. 340. http://milanesa.ime.usp.br/rbie/index.php/wcbie/article/view/6947/0

[8] Wilson de Jesus Masola and Norma Allevato. 2016. Dificuldades de aprendizagem matemática de alunos ingressantes na educação superior. *Revista Brasileira de Ensino Superior* 2, 1, 64–74. https://doi.org/10.18256/2447-3944/rebes.v2n1p64-74

[9] Paulo Vinícius Pereira de Lima, Geraldo Eustáquio Moreira, Lygianne Batista Vieira, and Maria Isabel Ramalho Ortigão. 2020. Brasil no Pisa (2003-2018): reflexões no campo da Matemática. *TANGRAM-Revista de Educação Matemática* 3, 2, 03–26. https://doi.org/10.30612/tangram.v3i2.12122

[10] Jeff Gothelf. 2012. Using Proto-Personas for Executive Alignment. https://uxmag.com/articles/using-proto-personas-for-executive-alignment. Accessed: 2023-10-25.

[11] Thiago Vinicius Jorge and Elvio Carlos Da Costa. 2021. Análise das modalidades de contratações CLT e PJ para os profissionais de tecnologia da informação. *Revista Interface Tecnológica* 18, 2, 91–104. https://doi.org/10.31510/infa.v18i2.1203

[12] Rosana Maria Luvezute Kripka, Moacir Kripka, Paolo Cezar de Nardin Pandolfo, Luiz Henrique Ferraz Pereira, Lori Viali, and Regis Alexandre Lahm. 2017. Aprendizagem de Álgebra Linear: explorando recursos do GeoGebra no cálculo de esforços em estruturas. *Revista Acta Scientiae* 19, 4, 544–562.

[13] S.J. Leon, E.A. Herman, and R. Faulkenberry. 1996. *ATLAST: Computer Exercises for Linear Algebra*. Prentice Hall.

[14] Francisco José de Lima and Joyce Pereira Oliveira. 2023. Desafios para a permanência no Ensino Superior: o caso de alunos ingressantes em um curso de licenciatura em matemática. *Revista Internacional de Educação Superior* v10. https://doi.org/10.20396/riesup.v10i00.8667417

[15] Ivo F Lopez and Claudia Segadas. 2014. A disciplina cálculo i nos cursos de engenharia da UFRJ: sua relação com o acesso à universidade e sua importância para a conclusão do curso. *Revista de Engenharia da Universidade Católica de Petrópolis* 8, 2, 92–107.

[16] Josué Antunes de Macêdo, Samara Neves de Almeida, and Marcos Rincon Voelzke. 2016. Descrições de programas livres e gratuitos para o ensino da Matemática. *Abakós* 4, 2, 3–19. https://doi.org/10.5752/P.2316-9451.2016v4n2p3

[17] Carmen Vieira Mathias. 2023. O potencial do GeoGebra como ferramenta de auxílio as habilidades de visualização. *Revista do Instituto GeoGebra Internacional de São Paulo* 12, 2, 044–066. https://doi.org/10.23925/2237-9657.2023.v12i2p044-066

[18] Diogo Chadud Milagres. 2018. AlgLin: uma ferramenta para mediar o processo de ensino-aprendizagem em álgebra linear. *Anais do XII Seminário Sul-Mato-Grossense de Pesquisa em Educação Matemática* 12, 1, 268–273.

[19] Fagner Alexandre Sotorriva Neckel. 2019. Geometria analítica e álgebra linear: A utilização do geogebra como ferramenta de ensino. *Revista Educação Online* 14, 30. https://doi.org/10.36556/eol.v14i30.418

[20] José Oliveira, Elaine Oliveira, Isabela Gasparini, and Leandro Wives. 2022. Repensando o ensino de computação. In *Anais Estendidos do II Simpósio Brasileiro de Educação em Computação* (Online). SBC, Porto Alegre, RS, Brasil, 16–16. https://doi.org/10.5753/educomp_estendido.2022.19400

[21] Stack Overflow. 2021. Stack Overflow Developer Survey 2021. https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies

[22] Roger S. Pressman and Bruce Maxim. 2014. *Software Engineering: A Practitioner's Approach* (8ª edição ed.). McGraw-Hill Science/Engineering/Math, New York, NY.

[23] Andriceli Richit, Maria Margarete Farias, Rosana Giaretta Miskulin, and Lêda Ferreira. 2013. Articulação entre álgebra linear e tecnologias digitais: perspectivas de exploração matemática no software GeoGebra. In *Actas del VII CIBEM* (Montevideo, Uruguay). SEMUR.

[24] Peter Shirley, Michael Ashikhmin, and Steve Marschner. 2009. *Fundamentals of computer graphics*. AK Peters/CRC Press.

[25] Thiago Ribeiro da Silva, Wesley Lucas Fernandes dos Santos, Ewerton Roosewelt Bernardo da Silva, and Alisson Werner Arruda de Arruda. 2021. Criação e usos do aplicativo LineAlg como objeto de aprendizagem na Educação Básica. *Diversitas Journal* 6, 1, 1415–1427. https://doi.org/10.17648/diversitas-journal-v6i1-1596

[26] Gaurav Sinha, Rahul Shahi, and Mani Shankar. 2010. Human Computer Interaction. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology* (Goa, India). 1–4. https://doi.org/10.1109/ICETET.2010.85

[27] Sepideh Stewart, Sheldon Axler, Robert Beezer, Eugene Boman, Minerva Catral, Guershon Harel, Judith McDonald, David Strong, and Megan Wawro. 2022. The linear algebra curriculum study group (LACSG 2.0) recommendations. *Notices of the American Mathematical Society* 69, 5, 813–819. https://doi.org/10.1090/noti2479

[28] Gaël Varoquaux, Lars Buitinck, Gilles Louppe, Olivier Grisel, Fabian Pedregosa, and Andreas Mueller. 2015. Scikit-learn: Machine learning without learning the machinery. *GetMobile: Mobile Computing and Communications* 19, 1, 29–33. https://doi.org/10.1145/2786984.2786995