PlayData: Programação em blocos para visualização de dados

Cassia Fernandez[1], João Adriano Freitas, Paulo Blisktein[2], Roseli de Deus Lopes[3]

[1]Unidade Acadêmica de Engenharia e Ciência da Computação – Insper
São Paulo, SP – Brazil

[2]Transformative Learning Technologies Lab, Teachers College – Columbia University
New York, U.SA.

[3]Depto. de Sistemas Eletrônicos, Escola Politécnica – Universidade de São Paulo (USP)
São Paulo, SP – Brazil

cassia.of1@insper.edu.br, jaafreitas@gmail.com,
paulob@tc.columbia.edu, roseli.lopes@usp.br

Abstract. *PlayData is a block-based programming (BBP) environment designed to support middle and high school students in creating data visualizations. Rooted in the constructionist learning theory, PlayData bridges the gap between the expressive freedom of general-purpose programming environments and the abstractions needed for easier working with data in BBP environments. Developed as an extension of Scratch 3.0, it introduces domain-specific blocks that enable users to import, manipulate, and visualize datasets with high transparency and low entry barriers. This paper presents the theoretical motivations, design principles, and technical features behind PlayData alongside examples that illustrate how the tool facilitates non-canonical visualizations to reflect learners' personal interests and goals. We conclude with reflections on current limitations and directions for future development.*

Resumo. *O PlayData é um ambiente de programação baseado em blocos (PBB) que busca apoiar estudantes do Ensino Fundamental II e Médio na criação de visualizações de dado. Fundamentado na teoria construcionista, o PlayData conecta a liberdade expressiva oferecida por ambientes de programação às abstrações necessárias para facilitar o trabalho com dados em ambientes de PBB. Desenvolvido como uma extensão do Scratch 3.0, a ferramenta introduz blocos específicos que permitem aos usuários importar, manipular e visualizar conjuntos de dados com alta transparência e baixas barreiras de entrada. Este artigo apresenta as motivações teóricas, os princípios de design e as características técnicas do PlayData, bem como exemplos que ilustram como a ferramenta facilita a criação de visualizações não canônicas para refletir interesses e objetivos pessoais dos estudantes. Concluímos com reflexões sobre as limitações da ferramenta e direções para o seu desenvolvimento futuro.*

1. Introdução

A educação em ciência de dados é um campo em rápido crescimento, que vem ganhando a atenção de pesquisadores, educadores e *designers* de tecnologias. Nos últimos anos, uma vasta gama de ferramentas educacionais para trabalhar com dados emergiu (Fernandez et al., 2025; Israel-Fishelson et al., 2023; Pimentel et al., 2022). No entanto, a maioria das plataformas destinadas à construção de visualizações oferece aos usuários um conjunto predefinido de modelos padronizados (*templates*), a partir dos quais visualizações podem ser criadas com apenas alguns cliques. Isso automatiza o trabalho, mas obscurece o processo de mapear valores de dados para formas representacionais, além de limitar as possibilidades de criação de representações menos convencionais (Rubin, 2020).

Ecoando o argumento construcionista (Turkle & Papert, 1991), partimos do pressuposto de que a construção de artefatos pessoalmente significativos e compartilháveis pode promover uma aprendizagem mais profunda e maior engajamento dos estudantes, além de tornar suas ideias tangíveis (Ackermann, 1996). No entanto, no campo da visualização de dados, ainda são escassas as ferramentas que permitem posturas mais autorais e expressivas por parte dos estudantes.

Neste trabalho, apresentamos o PlayData, uma ferramenta de programação em blocos projetada para apoiar estudantes do Ensino Fundamental II e Médio na criação de visualizações de dados, de forma a promover a transparência, expressividade e iteração neste processo. Detalhamos a motivação e os princípios de *design* que orientaram seu desenvolvimento, juntamente com uma descrição do ambiente e exemplos de suas aplicações.

2. Visualização de dados na Educação Básica

As visualizações de dados podem apoiar a aprendizagem em uma ampla gama de contextos disciplinares. À medida que novos objetivos para representar dados surgem no campo profissional, como o foco em comunicação e autoexpressão, novas abordagens educacionais também ganham relevância. Conectar visualização de dados à programação pode ser um caminho promissor para atingir esses objetivos, dada a expressividade, flexibilidade e potencial de customização oferecidas pela programação. Ferramentas de programação permitem a invenção de novas formas de visualizar e interagir com dados, bem como com o desenvolvimento de competências meta-representacionais (diSessa et al., 1991; diSessa & Sherin, 2000). Essas competências, intimamente relacionadas ao que mais recentemente tem sido chamado de letramento em visualização de dados (Börner et al., 2019), referem-se à "capacidade de selecionar, produzir e usar produtivamente representações, mas também de criticar e modificar representações e até mesmo projetar representações completamente novas" (diSessa & Sherin, 2000, p. 386, tradução nossa). Assim, ferramentas de programação podem tanto expandir as possibilidades de *design* de visualizações quanto oferecer oportunidades para que estudantes criem artefatos que sejam interessantes e relevantes para eles.

No entanto, grande parte da pesquisa sobre visualização de dados em contextos da Educação Básica tem se concentrado no trabalho com um número bastante limitado de representações canônicas (Börner et al., 2016; diSessa & Sherin, 2000), como gráficos de barra e de setores, muitas vezes sem muito espaço para o engajamento com os dados por meio de processos mais expressivos e criativos. De uma perspectiva técnica, a

implementação de abordagens que privilegiem novas formas de engajamento com dados, para além daquelas tradicionalmente valorizadas no âmbito da "análise objetiva de dados", exige ferramentas projetadas com esses objetivos específicos em mente.

Em ambientes computacionais, é importante considerar como as características das ferramentas usadas para fins de visualização podem impactar as formas de pensar dos estudantes (Rubin, 2020). Por exemplo, pesquisas mostram que os estudantes podem ter dificuldades para interpretar gráficos quando as escalas não estão devidamente ajustadas para revelar tendências específicas (Ben-Zvi & Arcavi, 2001; Ingulfsen et al., 2018). Nesse sentido, a escolha das ferramentas de visualização pode moldar significativamente a capacidade dos estudantes de extrair *insights* das informações apresentadas e impactar a forma como se envolvem em tarefas de construção de visualizações de dados.

3. Framework teórico

Embora na programação tradicional o código seja escrito em formato textual, na Educação Básica linguagens de programação baseadas em blocos (PBB) são a abordagem mais popular adotada para o ensino de programação. Em ferramentas de PBB, os usuários organizam conjuntos de blocos na tela arrastando-os e soltando-os, de modo semelhante ao encaixe de peças de blocos de montar. Erros de sintaxe são praticamente inexistentes, pois os blocos só podem ser organizados em configurações predefinidas de acordo com o seu formato, e os usuários podem ajustar o comportamento desses blocos modificando seus argumentos. Normalmente, estes ambientes possuem uma área dedicada na tela onde os usuários podem observar os resultados da execução do código. Exemplos populares de ambientes de PBB incluem o Scratch (Resnick et al., 2009), Snap! (Harvey et al., 2013), e Blockly. Estas ferramentas têm sido descritas como mais intuitivas para iniciantes (Weintrop & Wilensky, 2015) e, como os blocos descrevem diretamente as ações que executam e podem ser facilmente reorganizados, permitem que os aprendizes se envolvam com a programação de forma mais exploratória.

Em linguagens textuais, dois enfoques têm sido descritos para a visualização de dados: imperativo e declarativo. Em gramáticas **imperativas**, o código é escrito de modo a dizer ao computador exatamente *o que* fazer e *como* fazer, como uma sequência de passos. O Scratch pode ser considerado um ambiente de PBB imperativo, onde todos os passos devem ser explicitamente especificados, e a ordem dos blocos impacta diretamente os resultados. Essa abordagem oferece aos usuários liberdade para projetar diferentes tipos de visualizações e formas de interação com os dados, permitindo a construção de uma ampla gama de representações de dados (como animações, músicas, e outras formas artísticas). Contudo, implica em um custo cognitivo, com curvas de aprendizagem mais íngremes e uma esforço maior para realizar mudanças nas especificações da visualização (Grammel et al., 2013), o que pode tornar o processo de programação trabalhoso e complexo, especialmente para iniciantes (Heer & Bostock, 2010; Satyanarayan et al., 2020). Além disso, a ausência de métodos automatizados para criar legendas e eixos, por exemplo, pode dificultar a capacidade dos usuários de interpretá-los de maneiras precisas (Fernandez et al., 2024).

Gramáticas **declarativas,** no domínio da visualização de dados, correspondem a linguagens ou pacotes específicos para a construção de visualizações (Heer & Bostock, 2010; Mei et al., 2018). Elas diferem da programação imperativa por se concentrarem em declarar os *resultados desejados*, em vez de instruções diretas sobre como atingi-los. Isso

é alcançado por meio de linguagens mais concisas, nas quais as especificações (como o mapeamento de dados para eixos ou cores, por exemplo) são definidas como parâmetros de funções já prontas. Exemplos incluem o Plotly, uma biblioteca declarativa de gráficos para Python, e o BlockPy, um ambiente baseado em blocos em Python voltado para educação em ciência de dados. Como essas gramáticas buscam oferecer uma forma mais eficiente de construir visualizações, apenas um conjunto limitado de tipos de gráficos e parâmetros de personalização é disponibilizado e, portanto, as possibilidades de *design* costumam ser mais restritas (Satyanarayan et al., 2014).

Ferramentas educacionais de visualização de dados variam em relação a como apoiam diferentes objetivos de aprendizagem e formas de engajamento do usuário. Pesquisas mostram que o *design* das ferramentas impacta não apenas os tipos de visualizações que os usuários criam, mas também como eles pensam e aprendem com dados (Bostock & Heer, 2009; Méndez et al., 2017; Parsons et al., 2021). Frameworks clássicos para o desenvolvimento de ferramentas de visualização de dados enfatizam expressividade, eficiência e acessibilidade (Bostock & Heer, 2009), enquanto taxonomias mais recentes também consideram dimensões como orientação, usabilidade, colaboração e flexibilidade (Amini et al., 2018; Ren et al., 2019). Em contextos educacionais, essas dimensões são frequentemente reinterpretadas por meio das metáforas de *pisos, tetos e paredes* (Myers et al., 2000; Resnick et al., 2005), que se referem a quão fácil é começar a usar uma ferramenta (pisos baixos), quão longe ela pode levar os aprendizes (tetos altos) e quantos caminhos diferentes ela suporta (paredes amplas). Com base na literatura da área, em um trabalho anterior propusemos um *framework* para analisar ferramentas educacionais voltadas à visualização de dados, baseado em quatro dimensões centrais, detalhadas a seguir (Fernandez, Freitas, et al., 2025).

Abstração reflete o quanto de esforço é necessário para criar uma visualização inicial. Ferramentas mais abstratas (por exemplo, sistemas baseados em blocos com menos etapas) podem simplificar a criação, mas limitam a flexibilidade. Por outro lado, ferramentas com menor nível de abstração oferecem mais controle, porém exigem mais esforço e expertise por parte dos usuários.

Expressividade refere-se à diversidade de visualizações que uma ferramenta pode produzir. Alta expressividade permite formas visuais personalizadas ou inéditas, apoiando a criatividade, mas geralmente vem acompanhada de maior complexidade (ou menor abstração).

Transparência nos mapeamentos de dados indica o quão claramente uma ferramenta mostra a conexão entre os dados e os elementos visuais utilizados para representá-los. Ferramentas mais transparentes tornam esses mapeamentos explícitos, favorecendo a aprendizagem conceitual e a transferência para outras situações. Ferramentas do tipo "caixa-preta" ocultam esse processo, facilitando a criação, mas limitando a compreensão do que está por trás dos algoritmos.

Objetivos apoiados: A construção de visualizações de dados pode estar associada a múltiplos objetivos. Com base em classificações anteriores na área (Grammel et al., 2013; Masud et al., 2010; Satyanarayan et al., 2020), sintetizamos esses objetivos em duas categorias amplas: exploração e comunicação, entendendo-os como complementares e não hierárquicos. A exploração visa gerar *insights* por meio da descoberta de padrões e relações, e é favorecida por ferramentas de *feedback* rápido, mas que em geral oferecem menos possibilidades de personalização. Já a comunicação envolve expressar ideias,

perspectivas, provocar reflexão, e até produzir arte baseada em dados. Ferramentas voltadas a objetivos comunicativos oferecem altos níveis de customização (como escolha de cores, inclusão de anotações, destaque de pontos específicos, recursos para contar histórias e opções interativas), mas nem sempre facilitam a exploração.

4 PlayData

A maioria das linguagens de PBB desenvolvidas para fins de visualização de dados é projetada usando paradigmas declarativos. Embora variem no nível de personalização oferecido, seus resultados geralmente são semelhantes aos criados em editores baseados em *templates* (como Google Planilhas ou Excel), e o algoritmo faz escolhas ocultas relacionadas a conversões, mapeamentos visuais e aparência geral da representação. Essa abordagem permite que usuários criem gráficos tradicionais mais rapidamente, mas realiza transformações que podem parecer "mágicas" para os usuários.

Por outro lado, ferramentas de PBB mais gerais, como o Scratch, oferecem maior potencial de customização, ao custo de uma menor abstração e, portanto, maior complexidade. Assim, os ambientes existentes geralmente oferecem ou baixa abstração ou baixa expressividade. O PlayData adota uma abordagem que combina as capacidades expressivas do Scratch com a abstração de alguns de seus blocos, ao mesmo tempo em que incorpora novos recursos que facilitam o *design* de visualizações. O objetivo é manter alta transparência nos mapeamentos de dados sem sacrificar a expressividade, alinhando a ferramenta de maneira mais próxima aos objetivos comunicativos (atualmente pouco atendidos por outras ferramentas).

4.1 Princípios norteadores do PlayData

O PlayData é uma linguagem para visualização de dados no Scratch 3.0, direcionada a estudantes do Ensino Fundamental II e Médio[1]. A ferramenta é inspirada na teoria da aprendizagem construcionista desenvolvida por Seymour Papert e colegas (Papert & Harel, 1991), que propõe que a construção de conhecimento é facilitada quando os aprendizes têm a oportunidade de criar e compartilhar artefatos pessoalmente significativos. Papert também destacou a importância de se ter boas ferramentas "para pensar" durante o processo de aprendizagem. Alinhado a essa ideia, o PlayData foi idealizado para permitir que aprendizes se envolvessem em tarefas relacionadas a dados por meio de processos exploratórios, onde, munidos de ferramentas apropriadas, eles "podem construir, testar, investigar e experimentar sem perguntas ou instruções impostas externamente" (Hawkins, 1965).

O construcionismo tem uma longa história de tornar ideias complexas acessíveis a crianças. Essa tradição, visível em ferramentas como o Logo (Papert, 1980), StarLogo (Resnick, 1999), Scratch (Resnick et al., 2009), e diversos kits de robótica (Blikstein, 2015), envolve selecionar conceitos ou formas de pensar relevantes de uma determinada área e traduzi-los para contextos educacionais. Ferramentas fundamentadas no construcionismo são tipicamente projetadas como "objetos-para-pensar-com", permitindo novas formas de raciocínio (Ackermann, 2001; Papert, 1980), ao mesmo tempo em que oferecem experiências de aprendizagem pessoalmente significativas.

[1] O PlayData está disponível gratuitamente em https://playdatalab.github.io/.

Portanto, tanto aspectos epistemológicos quanto pessoais desempenham um papel central no *design* de tecnologias construcionistas (Resnick et al., 1996).

Alinhado a essas ideias, o desenvolvimento do PlayData foi guiado por uma abordagem de pesquisa baseada em design (Barab & Squire, 2004; DBRC, 2003), fundamentada na conjectura de alto nível (Sandoval, 2014) de que *a construção de uma visualização pessoalmente significativa e compartilhável pode ampliar a participação em atividades relacionadas a dados e apoiar o processo de construção de sentido dos aprendizes sobre dados.* Ao falarmos em "construção", queremos dizer que os estudantes precisam traduzir intencionalmente uma ideia em uma visualização, tomando decisões sobre como codificar os dados em formas representacionais, em vez de simplesmente seguir sugestões feitas pelo sistema. Isso pode promover um senso de autoria no processo de aprendizagem e fomentar uma compreensão mais profunda sobre como os dados são mapeados em atributos visuais. Quando nos referimos a artefatos "pessoalmente significativos", enfatizamos que os estudantes podem criar visualizações personalizadas que refletem suas próprias decisões de design e possuem significado pessoal. Essa customização pode potencialmente empoderar os aprendizes a criar visualizações que ressoem com suas perspectivas, perguntas e percepções únicas, estabelecendo conexões mais fortes com os dados. Por fim, ao construir um artefato "compartilhável", os estudantes podem se engajar com visualização de dados como uma tarefa comunicativa, tomando decisões não apenas sobre a aparência geral de suas criações, mas também sobre os aspectos específicos dos dados que desejam enfatizar.

4.2 Implementação do PlayData

O ambiente Scratch

O Scratch é uma linguagem de programação baseada em blocos desenvolvida pelo grupo *Lifelong Kindergarten* do MIT Media Lab para que crianças e jovens criem seus próprios jogos, histórias interativas e animações (Resnick et al., 2009). O Scratch é também uma comunidade online com milhões de usuários ao redor do mundo, onde podem compartilhar seus projetos e interagir uns com os outros. Os blocos do Scratch 3.0 são baseados no Blockly, uma biblioteca para a criação de editores visuais de código para sites e aplicativos. Como o Scratch 3.0 é de código aberto, desenvolvedores podem estender o ambiente criando blocos personalizados e outros recursos que executam funções específicas. Os principais elementos do ambiente de programação do Scratch são a paleta de blocos, a área de programação (onde os blocos são combinados para criar programas), a área do palco (onde a saída do programa é visualizada), e o menu de atores e cenários. Ele também oferece ferramentas de edição visual e sonora, acessíveis em abas específicas. No Scratch, os blocos são arrastados da paleta de blocos e combinados na área de programação por meio de mecanismos de encaixe.

O ambiente PlayData

O PlayData foi projetado para manter um alinhamento com a sintaxe e os princípios de design dos blocos originais do Scratch, incluindo: (i) manter consistência com os termos e expressões usados nos blocos do Scratch; (ii) evitar erros de sintaxe e mensagens de erro; (iii) evitar um número excessivo de parâmetros dentro dos blocos, bem como manter o número de blocos restrito ao menor número possível; e (iv) evitar duplicar blocos que executem funções semelhantes às já oferecidas pelo Scratch.

Na primeira fase do design da ferramenta, identificamos as principais funções necessárias para criar visualizações de dados que são geralmente difíceis de realizar no Scratch. Quatro funções foram consideradas centrais nesta etapa:

1. Importar conjuntos de dados de fontes externas,
2. Iterar pelos valores do conjunto de dados para lê-los,
3. Alterar a escala dos valores do conjunto de dados para mapear diferentes faixas,
4. Visualizar o conjunto de dados no ambiente em uma tabela.

Outras funções surgiram posteriormente, após testes iniciais com estudantes, principalmente para permitir a realização de diferentes tipos de análise de dados sem a necessidade de sair do ambiente (como filtragem de dados e cálculo de estatísticas básicas e agregações). Destacamos que um dos principais desafios no *design* da ferramenta foi equilibrar a variedade de blocos necessários para executar diferentes funções que possibilitam a criação de visualizações e análises (alta expressividade), mantendo ao mesmo tempo o número de blocos e parâmetros o mais baixo possível.

As principais diferenças do PlayData em relação ao ambiente original do Scratch (Figura 1) são: (1) novos blocos para trabalhar com dados; (2) uma tabela embutida no ambiente, que permite aos usuários visualizar rapidamente o conjunto de dados, sendo aberta ao clicar no ícone da tabela (3); (4) atores e cenários personalizados que podem ser usados em projetos de visualização de dados (como círculos, quadrados e mapas). Além disso, quando um conjunto de dados é importado, os menus suspensos dentro dos blocos do PlayData são automaticamente atualizados (5).

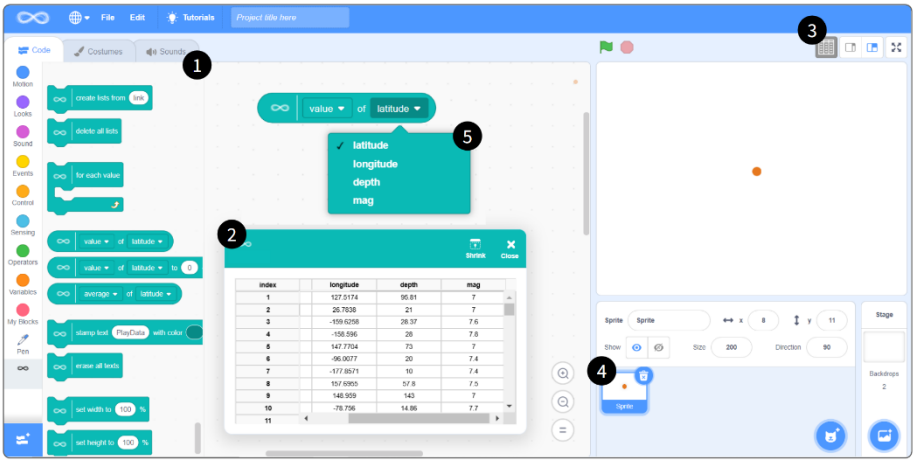

Figura 1. Visão geral do ambiente PlayData

Blocos do PlayData

Embora o Scratch permita tarefas de análise e visualização de dados por meio de seus blocos tradicionais, muitos usuários enfrentam dificuldades com o pensamento abstrato necessário para usar listas de forma eficaz (Aivaloglou & Hermans, 2016; Paparo et al., 2021; Webb & Rosson, 2013). O Scratch oferece altos níveis de expressividade, mas, como não foi intencionalmente projetado para a construção de visualizações de dados, este tipo de tarefa pode exigir um conhecimento mínimo considerável e um alto esforço de programação. Em contraste, linguagens de programação em blocos que utilizam abordagens declarativas oferecem um ponto de entrada mais baixo, porém à custa de

redução da expressividade. O PlayData adota um caminho intermediário, com novos blocos que encapsulam procedimentos técnicos, mas mantendo a integração com todo o ambiente Scratch, o que oferece espaço para a criação de uma ampla variedade de formas representacionais. Algumas das principais funcionalidades da ferramenta são descritas a seguir.

Iterando sobre os valores: No Scratch, para iterar sobre os valores de uma lista, os usuários precisam utilizar listas e uma variável contadora (que deve ser inicializada e incrementada dentro de um bloco de repetição até que toda a lista seja percorrida; Figura 2, à esquerda). Esse processo pode ser intuitivo para programadores, mas é complexo para iniciantes, exigindo conhecimentos sobre listas e variáveis que geralmente não lhes são familiares (Aivaloglou & Hermans, 2016). No PlayData, essas etapas são incorporadas em um bloco de laço que itera sobre o conjunto de dados (Figura 2, à direita), facilitando a sua leitura e uso para fins diversos.

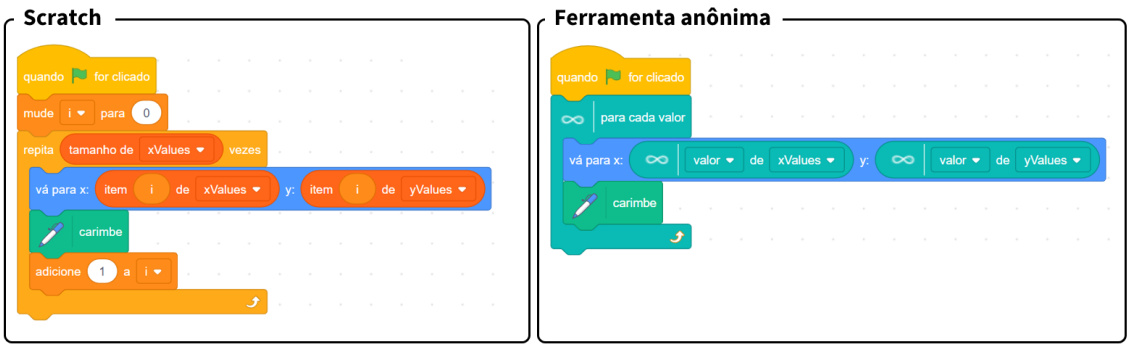

Figura 2: Comparação do código criado para iterar sobre uma lista de valores no Scratch (esquerda) e no PlayData (direita).

Alterando o intervalo: Um desafio significativo na criação de visualizações no Scratch é ajustar o intervalo original dos valores para que se adeque ao ambiente. Por exemplo, mapear dados para o eixo X-Y exige modificar os valores da lista para que se adaptem ao intervalo da tela (de –180 a 180 para o eixo Y e de –240 a 240 para o eixo X). Se forem mapeados para cores, os valores variam de 0 a 200. Para fins musicais, o intervalo de notas vai de 0 a 130. Converter os valores para esses intervalos exige a realização de cálculos usando diversos blocos, o que pode ser pouco intuitivo para iniciantes e trabalhoso. Para resolver esse problema, dois blocos no PlayData permitem que os valores das listas que compõem o conjunto de dados sejam mapeados automaticamente para qualquer intervalo desejado, facilitando uma exploração mais rápida e o refinamento iterativo das representações (Figura 3).

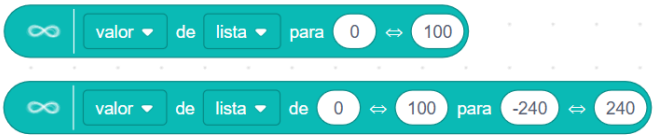

Figura 3. Blocos que mapeiam o intervalo de saída com base nos valores máximo e mínimo da lista definidos automaticamente (acima) e manualmente (abaixo)

Uma comparação dos blocos necessários para realizar transformações de intervalo no Scratch e no PlayData é apresentada na Figura 4.

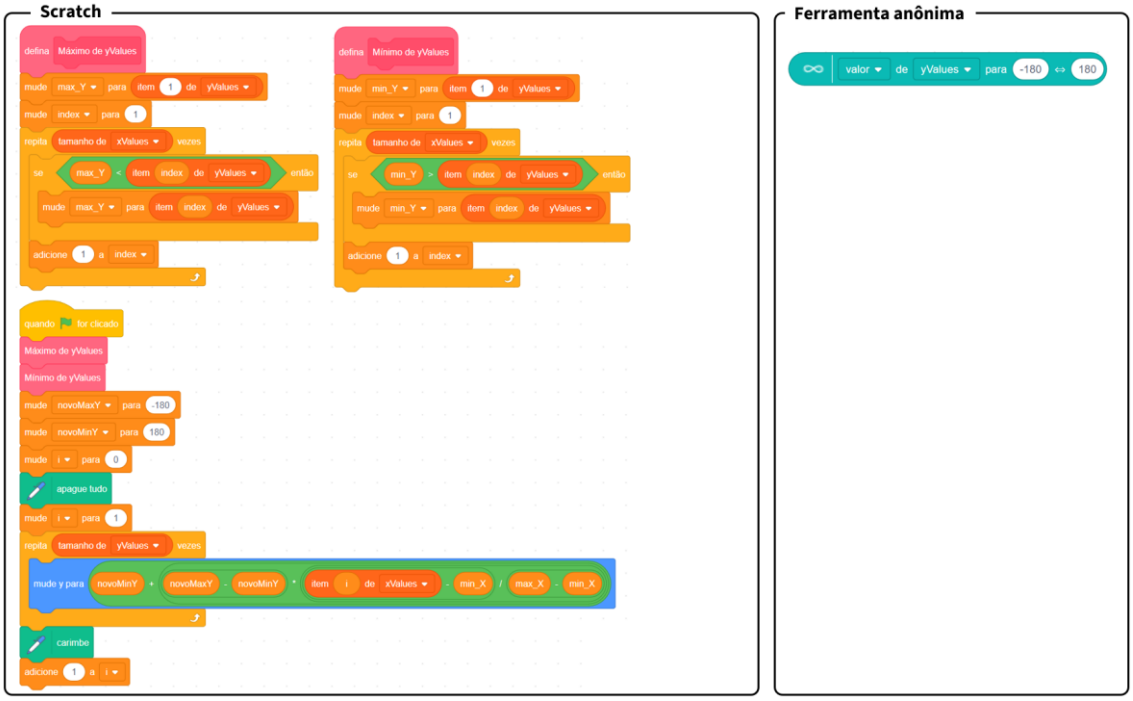

Figura 4. Comparação dos blocos necessários para mapear os valores máximos e mínimos para um intervalo específico no Scratch (esquerda) e PlayData (direita)

Transformando dados: Cálculos simples (como o valor mínimo, máximo ou médio de uma lista) podem exigir uma combinação de blocos de listas, variáveis e operadores no Scratch. O PlayData abstrai esses cálculos em novos blocos (Figura 5), além de incluir blocos que permitem filtrar, ordenar e agrupar dados com base em critérios específicos.

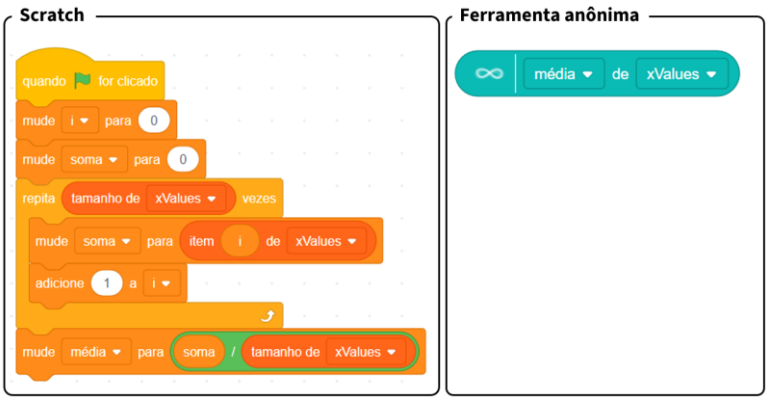

Figura 5. Código para calcular o valor médio de uma lista com os blocos do Scratch e do PlayData

Importando conjuntos de dados: Conjuntos de dados podem ser adicionados ao Scratch por meio da inserção manual de valores nos blocos de lista ou da importação de um arquivo CSV armazenado localmente no computador. Para oferecer formas mais rápidas e flexíveis de importar conjuntos de dados, o PlayData apresenta um bloco para importar dados de planilhas online ou conjuntos de dados CSV disponíveis online. Esse processo cria automaticamente novas listas que podem ser selecionadas nos menus suspensos

(como mostrado na Figura 6). Um bloco alternativo permite que os usuários adicionem dados colando valores copiados diretamente de tabelas ou planilhas.

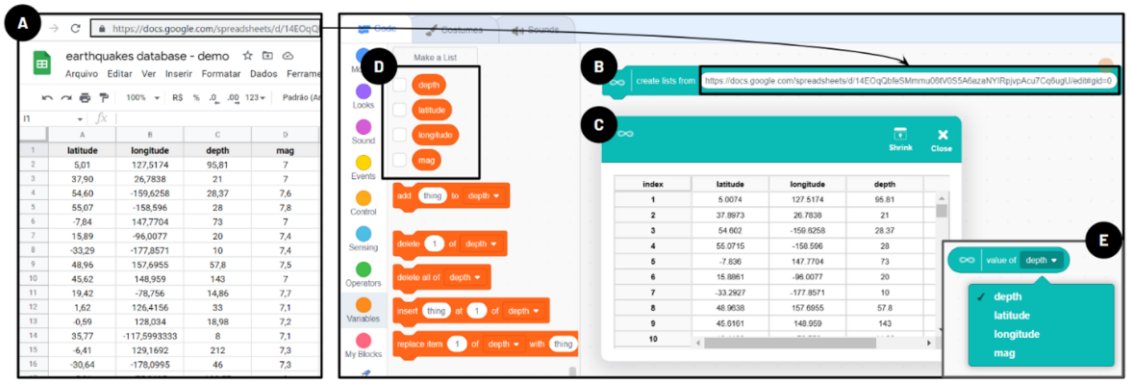

Figura 6. Exemplo de um conjunto de dados no Google Planilhas (A) sendo importado para o PlayData (B), os valores visualizados na tabela do ambiente (C), listas automaticamente adicionadas na paleta de blocos (D) e nos menus dos blocos (E).

Conjuntos de dados e estrutura de dados no PlayData

Quando um conjunto de dados é importado para o PlayData, listas são criadas automaticamente para cada uma de suas colunas. Elas podem então ser usadas como listas tradicionais do Scratch e aparecem automaticamente tanto na paleta de listas do Scratch quanto nos menus suspensos dentro dos blocos do PlayData. As listas importadas podem armazenar tanto números quanto texto. Quando o bloco de laço é utilizado, o programa itera sobre cada linha do conjunto de dados e retorna o valor da lista selecionada (coluna do conjunto de dados), um por um. Um exemplo é apresentado na Figura 7.

Figura 7. Exemplo de dados sendo representados na tela de saída

Optamos por permitir que os usuários trabalhem com apenas um conjunto de dados (tabela) por vez. Essa decisão foi tomada para reduzir a complexidade para iniciantes, evitando a necessidade de incluir outro parâmetro dentro dos blocos do PlayData para definir o conjunto de dados associado.

5. Demonstração de aplicações

Nesta seção, apresentamos exemplos de códigos e saídas criados com o PlayData para representar dados de diferentes naturezas e as possibilidades expressivas da ferramenta. Na Figura 8, apresentamos um exemplo de um mapa criado com o PlayData em que terremotos são plotados temporalmente, como uma animação, com base em sua magnitude (tamanho), profundidade (cor) e localização (dados geoespaciais), juntamente com o código associado.

Figura 8. Resultado final de uma animação que mostra terremotos de acordo com suas posições, magnitude (tamanho) e profundidade (cor).

Na Figura 9, utilizando o mesmo conjunto de dados da Figura 8, os terremotos são plotados com base em sua profundidade (eixo Y) e magnitude (tamanho e cor).

Figura 9. Visualização de terremotos ao longo do tempo de acordo com suas posições, magnitude (tamanho) e profundidade (cor).

A combinação do PlayData com os blocos da extensão de música do Scratch permite a criação de recursos de sonificação que proporcionam uma experiência multissensorial da variabilidade de um conjunto de dados. Na Figura 10, apresentamos um exemplo de código que mapeia valores de um conjunto de dados para notas musicais.

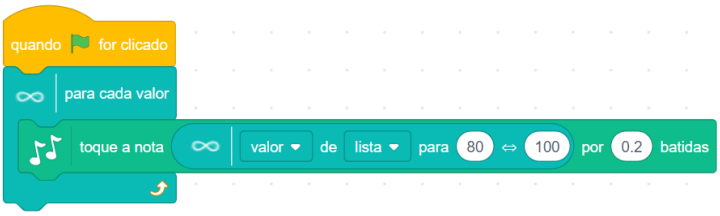

Figura 10. Combinação do PlayData com blocos da extensão de música, mostrando como dados podem ser mapeados em diferentes notas.

6. Conclusão e trabalhos futuros

Atualmente, o PlayData apresenta potencialidades, mas também desafios para tarefas de visualização de dados. Um desafio relacionado à abordagem de programação imperativa adotada no design da ferramenta é o potencial de erros na interpretação das visualizações. Por exemplo, como eixos e legendas não são gerados automaticamente, os usuários podem interpretar incorretamente o intervalo de dados exibidos, o que pode levar a conclusões equivocadas (Fernandez et al., 2024). Além disso, como a ferramenta se concentra em objetivos de comunicação, ela pode não ser tão eficaz para apoiar a exploração rápida de dados, já que é necessário um esforço maior para criar uma visualização inicial.

À medida que observamos o seu uso por estudantes em salas de aula reais de contextos diversos, temos a intenção de seguir refinando o seu design a partir dos desafios que emergem. Atualmente, planejamos incluir algumas funcionalidades específicas como próximos passos, incluindo: a integração com dispositivos de computação física para acessar dados coletados do ambiente de forma contínua; a integração com serviços de armazenamento (por exemplo, Google Drive) para salvar e carregar projetos da nuvem; desenvolvimento de tutoriais embutidos para orientar usuários iniciantes e inspirar novas formas de uso da ferramenta; e desenvolvimento de novos blocos para realizar mapeamentos de dados com base em valores categóricos.

Por fim, destacamos que o engajamento produtivo com visualizações de dados requer a combinação de diversas práticas e conhecimentos (Fernandez, Blikstein, et al., 2025; H. S. Lee et al., 2022; V. R. Lee & Wilkerson, 2018), que podem ser desenvolvidos por meio do uso de diferentes ferramentas. Nesse sentido, vemos o PlayData como uma opção adicional dentro de um conjunto mais amplo de ferramentas que podem se complementar e apoiar os estudantes a desenvolver o letramento em dados de formas mais significativas e pessoais.

Uso de Inteligência Artificial

Utilizamos o ChatGPT para revisar a gramática de partes específicas do texto e traduzir trechos do artigo que constavam em uma tese de doutorado redigida em inglês. Após o uso da ferramenta, esses trechos foram revisados e refinados pelos autores para gerar a versão final do texto.

Referências

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*.

Ackermann, E. (1996). Perspective-Taking and Object Construction: Two Keys to Learning. *Constuctionism in Practice: Designing, Thinking, and Learning in a Digital World*, 25–37.

Aivaloglou, E., & Hermans, F. (2016). How kids code and how we know: An exploratory study on the Scratch repository. *ICER 2016 - Proceedings of the 2016 ACM Conference on International Computing Education Research*, 53–61. https://doi.org/10.1145/2960310.2960325

Amini, F., Bolduan, M. M. B., Elmer, C., & Wiederkehr, B. (2018). Evaluating Data-Driven Stories and Storytelling Tools. In A. Thudt, J. Walny, T. Gschwandtner, J. Dykes, & J. Stasko (Eds.), *Data-Driven Storytelling* (pp. 249–286). Taylor & Francis/CRC Press. https://doi.org/10.1201/9781315281575-3

Barab, S., & Squire, K. (2004). Design-Based Research: Putting a Stake in the Ground. *The Journal of the Learning Sciences*, *13*(1), 1–14. https://doi.org/10.1207/s15327809jls1301_1

Ben-Zvi, D., & Arcavi, A. (2001). Junior high school students' construction of global views of data and data representations. *Educational Studies in Mathematics*, *45*(1–3), 35–65. https://doi.org/10.1023/A:1013809201228

Blikstein, P. (2015). Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Foundations and Trends® in Human–Computer Interaction*, *9*(1), 1–68. https://doi.org/10.1561/1100000057

Börner, K., Bueckle, A., & Ginda, M. (2019). Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proceedings of the National Academy of Sciences of the United States of America*, *116*(6), 1857–1864. https://doi.org/10.1073/pnas.1807180116

Börner, K., Maltese, A., Balliet, R. N., & Heimlich, J. (2016). Investigating aspects of data visualization literacy using 20 information visualizations and 273 science museum visitors. *Information Visualization*, *15*(3), 198–213. https://doi.org/10.1177/1473871615594652

Bostock, M., & Heer, J. (2009). Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, *15*(6), 1121–1128. https://doi.org/10.1109/TVCG.2009.174

DBRC, The Design-Based Research Collective (2003). Design-based research: An emerging paradigm for educational inquiry. *Educational Research*, *32*(1), 5–8. https://doi.org/10.3102/0013189X032001005

diSessa, A. A., Hammer, D., Sherin, B. L., & Kolpakowski, T. (1991). Inventing Graphing: Meta-Representational Expertise in Children. *Journal of Mathematical Behavior*, *10*, 117–160. http://eric.ed.gov/ERICWebPortal/detail?accno=EJ443633

diSessa, A. A., & Sherin, B. L. (2000). Meta-representation: An introduction. *Journal of Mathematical Behavior*, *19*(4), 385–398. https://doi.org/10.1016/S0732-3123(01)00051-7

Fernandez, C., Blikstein, P., & Lopes, R. D. D. L. (2024). Design failures in data visualization programming activities. *Proceedings of ACM Interaction Design and Children Conference: Inclusive Happiness, IDC 2024*, 574–586. https://doi.org/10.1145/3628516.3663348

Fernandez, C., Blikstein, P., & Lopes, R. de D. (2025). A Multi-Method Approach for Exploring Programming Trajectories Through Log Data: Insights from Data Visualization Tasks. *Journal of Science Education and Technology*. https://doi.org/10.1007/s10956-025-10210-7

Fernandez, C., Freitas, J. A., Blikstein, P., & de Deus Lopes, R. (2025). The design space of visualization tools for data science education: Literature review and framework

for future designs. *International Journal of Child-Computer Interaction*, *43*. https://doi.org/10.1016/j.ijcci.2024.100698

Grammel, L., Bennett, C., Tory, M., & Storey, M.-A. (2013). A Survey of Visualization Construction User Interfaces. *Eurographics Conference on Visualization (EuroVis)*, 19–23. https://doi.org/10.2312/PE.EuroVisShort.EuroVisShort2013.019-023

Harvey, B., Garcia, D. D., Barnes, T., Titterton, N., Armendariz, D., Segars, L., Lemon, E., Morris, S., & Paley, J. (2013). SNAP! (build your own blocks) (abstract only). *Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13*, 759. https://doi.org/10.1145/2445196.2445507

Hawkins, D. (1965). Messing about in Science. *Science and Children*, *2*(5).

Heer, J., & Bostock, M. (2010). Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, *16*(6), 1149–1156. https://doi.org/10.1109/TVCG.2010.144

Ingulfsen, L., Furberg, A., & Strømme, T. A. (2018). Students' engagement with real-time graphs in CSCL settings: scrutinizing the role of teacher support. *International Journal of Computer-Supported Collaborative Learning*, *13*(4), 365–390. https://doi.org/10.1007/s11412-018-9290-1

Israel-fishelson, R., Moon, P. F., Tabak, R., & Weintrop, D. (2023). Preparing K-12 Students to Meet their Data : Analyzing the Tools and Environments used in Introductory Data Science Contexts. *Learning, Design and Technology (LDT '23), June 23, 2023, Evanston, IL, USA. ACM, New York, NY, USA*, 29–42. https://doi.org/10.1145/3594781.3594796

Lee, H. S., Mojica, G. F., Thrasher, E. P., & Baumgartner, P. (2022). Investigating data like a data scientist: key practices and processes. *Statistics Education Research Journal*, *21*(2). https://doi.org/10.52041/serj.v21i2.41

Lee, V. R., & Wilkerson, M. (2018). *Data use by middle and secondary students in the digital age: A status report and future prospects. Commissioned Paper for the National Academies of Engineering, and Medicine, Board on Science Education, Committee on Science Investigations and Engineering*.

Masud, L., Valsecchi, F., Ciuccarelli, P., Ricci, D., & Caviglia, G. (2010). From data to knowledge: Visualizations as transformation processes within the data-information-knowledge continuum. *Proceedings of the International Conference on Information Visualisation*, 445–449. https://doi.org/10.1109/IV.2010.68

Mei, H., Ma, Y., Wei, Y., & Chen, W. (2018). The design space of construction tools for information visualization: A survey. *Journal of Visual Languages and Computing*, *44*, 120–132. https://doi.org/10.1016/j.jvlc.2017.10.001

Méndez, G. G., Hinrichs, U., & Nacenta, M. A. (2017). Bottom-up vs. Top-down: Trade-offs in efficiency, understanding, freedom and creativity with infovis tools. *Conference on Human Factors in Computing Systems - Proceedings*, *2017-May*, 841–852. https://doi.org/10.1145/3025453.3025942

Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, *7*(1), 3–28. https://doi.org/10.1145/344949.344959

Paparo, G., Hartmann, M., & Grillenberger, M. (2021). A Scratch Challenge: Middle School Students Working with Variables, Lists and Procedures. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3488042.3488065

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. http://dl.acm.org/citation.cfm?id=1095592

Papert, S., & Harel, I. (1991). Situating constructionism. In S. Papert & I. Harel (Eds.), *Constructionism*. Ablex Publishing Corporation.

Parsons, P., Shukla, P., & Park, C. (2021). Fixation and Creativity in Data Visualization Design: Experiences and Perspectives of Practitioners. *Proceedings - 2021 IEEE Visualization Conference - Short Papers, VIS 2021*, 76–80. https://doi.org/10.1109/VIS49827.2021.9623297

Pimentel, D. R., Horton, N. J., & Wilkerson, M. H. (2022). *Tools to Support Data Analysis and Data Science in K-12 Education*.

Ren, D., Lee, B., Brehmer, M., & Riche, N. H. (2019). Reflecting on the evaluation of visualization authoring systems. *Proceedings - 7th Biennial Workshop Evaluation and Beyond: Methodological Approaches for Visualization, BELIV 2018*, (October), 86–92. https://doi.org/10.1109/BELIV.2018.8634297

Resnick, M. (1999). Decentralized Modeling and Decentralized Thinking. In W. Feurzeig & N. Roberts (Eds.), *Modeling and Simulation in Precollege Science and Mathematics* (pp. 114–137). Springer.

Resnick, M., Bruckman, A., & Martin, F. (1996). Pianos not stereos: Creating Computational Construction Kits. *Interactions*, *3*(6), 40–50.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. a Y., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, *52*, 60–67.

Resnick, M., Myers, B., Nakakoji, K., Shneiderman, B., Pausch, R., Selker, T., & Eisenberg, M. (2005). Design Principles for Tools to Support Creative Thinking. In *Technical Report: NSF Workshop Report on Creativity Support Tools.*

Rubin, A. (2020). Learning to Reason with Data: How Did We Get Here and What Do We Know? *Journal of the Learning Sciences*, *29*(1), 154–164. https://doi.org/10.1080/10508406.2019.1705665

Sandoval, W. (2014). Conjecture Mapping: An Approach to Systematic Educational Design Research. *Journal of the Learning Sciences*, *23*(1), 18–36. https://doi.org/10.1080/10508406.2013.778204

Satyanarayan, A., Lee, B., Ren, D., Heer, J., Stasko, J., Thompson, J., Brehmer, M., & Liu, Z. (2020). Critical reflections on visualization authoring systems. *IEEE Transactions on Visualization and Computer Graphics*, *26*(1), 461–471. https://doi.org/10.1109/TVCG.2019.2934281

Satyanarayan, A., Wongsuphasawat, K., & Heer, J. (2014). Declarative interaction design for data visualization. *UIST 2014 - Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, 669–678. https://doi.org/10.1145/2642918.2647360

Turkle, S., & Papert, S. (1991). Epistemological Pluralism and the Revaluation of the Concrete. In *Constructionism* (pp. 161–191). Ablex Publishing Corporation.

Webb, H. C., & Rosson, M. B. (2013). Using scaffolded examples to teach computational thinking concepts. *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 95–100. https://doi.org/10.1145/2445196.2445227

Weintrop, D., & Wilensky, U. (2015). To block or not to block? That is the question: Students' Perceptions of Blocks-based Programming. *IDC '15: Proceedings of the 14th International Conference on Interaction Design and Children*, *149*(4), 199–208. https://doi.org/10.1016/j.jtcvs.2015.01.023