Pontifícia Universidade Católica de Minas Gerais

Curso: Engenharia de Software

Matéria: Laboratório de Experimentação de Software

Características de Repositórios Populares

Relatório Técnico

Autores:
Andre Hyodo
Gustavo Pereira

August 28, 2025

1 Introdução e Hipóteses

Neste estudo, analisamos repositórios populares do GitHub para responder questões relacionadas à contribuição externa, número de releases, frequência de atualização e linguagens de programação utilizadas.

Hipóteses (Hipótese Nula e Alternativa)

Para cada questão de pesquisa (RQ), formulamos hipóteses informais:

- **RQ01 – Idade dos repositórios** H_0: Repositórios populares não são necessariamente antigos. H_a: Repositórios populares tendem a ser mais maduros/antigos.
- **RQ02 – Contribuição externa (PRs aceitos)** H_0: Repositórios populares não recebem muitas contribuições externas. H_a: Repositórios populares recebem muitas contribuições externas.
- **RQ03 – Releases** H_0: Repositórios populares não lançam releases com frequência. H_a: Repositórios populares lançam releases com frequência.
- **RQ04 – Atualizações** H_0: Repositórios populares não são atualizados com frequência. H_a: Repositórios populares são atualizados frequentemente.
- **RQ05 – Linguagens** H_0: Repositórios populares não estão concentrados em linguagens amplamente utilizadas. H_a: Repositórios populares estão majoritariamente em linguagens mais populares.
- **RQ06 – Issues fechadas** H_0: Repositórios populares não apresentam alta taxa de issues fechadas. H_a: Repositórios populares apresentam alto percentual de issues fechadas.
- **RQ07 – Bônus (por linguagem)** H_0: A linguagem do repositório não influencia significativamente a quantidade de contribuições, releases ou frequência de atualização. H_a: Repositórios em linguagens populares recebem mais contribuições externas, lançam mais releases e são atualizados com mais frequência.

2 Metodologia

O processo de coleta e análise dos dados foi realizado em etapas, com ajustes ao longo do desenvolvimento devido a limitações técnicas da API do GitHub:

- **Tentativa inicial**: começamos utilizando apenas a API GraphQL para coletar os dados diretamente dos 1000 repositórios mais populares. Entretanto, a execução retornava erro *Bad Gateway*, impossibilitando a coleta em uma única consulta.
- **Refatoração**: para contornar esse problema, adotamos uma estratégia híbrida.
  - Utilizamos a **API REST** para obter apenas o nome e o número de estrelas dos repositórios, filtrando aqueles com mais de 10.000 estrelas, o que também reduziu o volume de dados e otimizou o código.

  - Com a lista de nomes dos 1000 repositórios mais populares, realizamos consultas específicas na **API GraphQL**, recuperando as demais métricas necessárias.

- **Métricas extraídas**: para cada repositório, coletamos:
  - Data de criação (para cálculo da idade).
  - Total de pull requests aceitas.
  - Número de releases.
  - Data da última atualização.
  - Linguagem primária.
  - Total de issues abertas e fechadas (para cálculo da razão).

- **Eficiência**: com essa abordagem, conseguimos reduzir o tempo total de execução para aproximadamente **245 segundos** para os 1000 repositórios.

- **Tratamento e análise**:
  - Calculamos os valores medianos para cada métrica.
  - Organizamos os repositórios por linguagem, destacando as mais populares.
  - Realizamos comparações entre linguagens populares e outras, respondendo às questões de pesquisa.

3 Resultados

Contagem por Categoria (Linguagem de Programação)

- Python: 187 repositórios
- JavaScript: 130
- TypeScript: 156
- Java: 50
- C#: 12
- C++: 48
- PHP: 7
- Shell: 19
- C: 25
- Go: 73
- Outras linguagens: 293

Valores Medianos por Linguagem Popular

Table 1: Valores Medianos por Linguagem Popular

Linguagem	PRs Aceitos	Releases	Última Atualização
Python	548	22	1.0 horas
JavaScript	533.5	33	3.0 horas
TypeScript	2103.5	146.5	1.0 horas
Java	645	42.5	3.0 horas
C#	2780.5	100	0.5 horas
C++	932	58	1.0 horas
PHP	2944	346	0.0 horas
Shell	437	2	4.0 horas
C	113	32	0.0 horas
Go	1677	124	2.0 horas

4 Discussão

Hipóteses e Expectativas

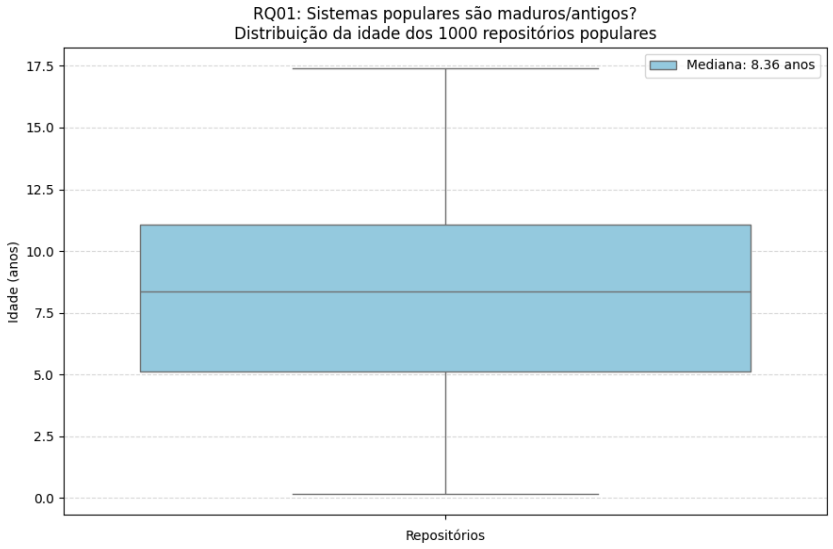
Para cada questão de pesquisa, estabelecemos hipóteses nula (H_0) e alternativa (H_a). De forma geral, esperávamos que repositórios populares apresentassem:

- Idade elevada, indicando maturidade do projeto (RQ01).
- Elevado número de contribuições externas (PRs aceitos) (RQ02).
- Maior número de releases (RQ03).
- Atualizações frequentes, com pouco tempo desde a última modificação (RQ04).
- Concentração em linguagens populares como Python, JavaScript e TypeScript (RQ05).
- Alta taxa de issues fechadas (RQ06).
- Para o bônus (RQ07), que linguagens populares apresentassem maior atividade em PRs, releases e atualizações do que linguagens menos utilizadas.

5 Resultados e Discussão

RQ01 – Idade dos repositórios

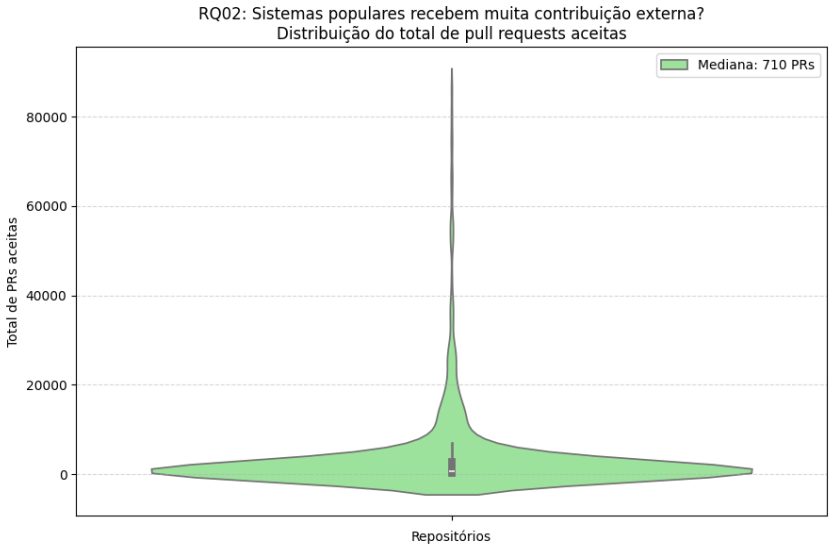
A análise da idade mostra que a maioria dos repositórios populares possui de 5 a 11 anos de existência, apoiando H_a de que são sistemas maduros.



Distribuição da idade dos repositórios (boxplot)

RQ02 – Contribuição externa (PRs aceitos)

As medianas de PRs aceitos variam bastante entre linguagens. Exemplos: Python (548), JavaScript (533,5), TypeScript (2103,5), Java (645), C# (2780,5), C++ (932), PHP (2944), Shell (437), C (113) e Go (1677). Esses valores confirmam H_a, mostrando que repositórios populares recebem, em geral, muitas contribuições externas.

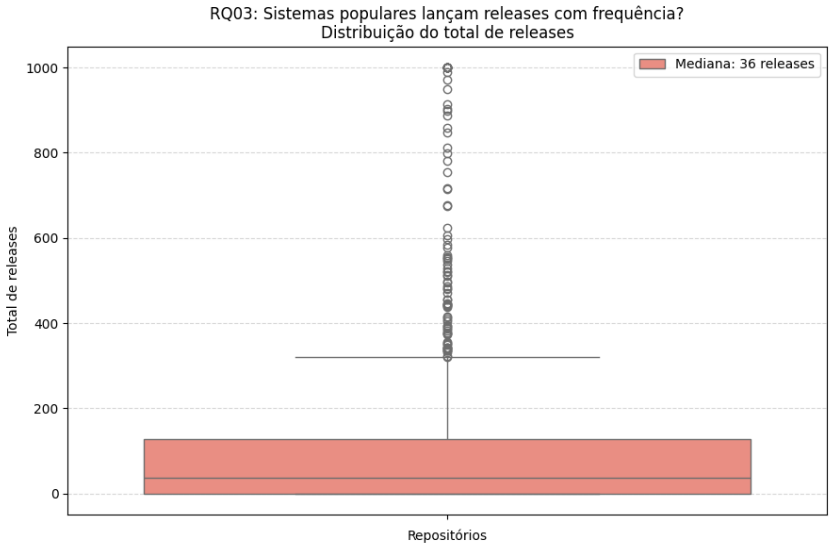


Distribuição do total de PRs aceitas (violinplot)

RQ03 – Releases

As medianas de releases também mostram disparidade. Destacam-se PHP (346), TypeScript (146,5) e Go (124), enquanto Shell aparece com apenas 2 releases. Isso apoia par-

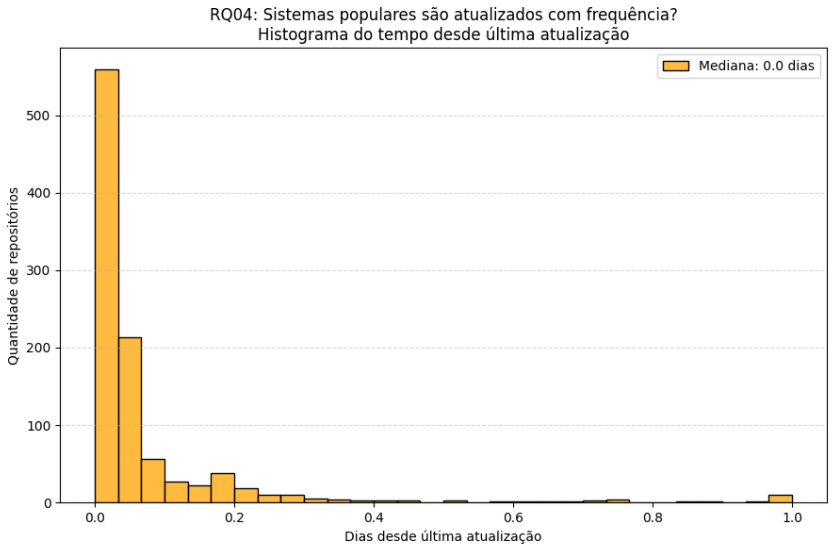
cialmente H_a: repositórios populares lançam releases com frequência, mas há variações significativas entre linguagens.



Distribuição do total de releases (boxplot)

RQ04 – Última atualização

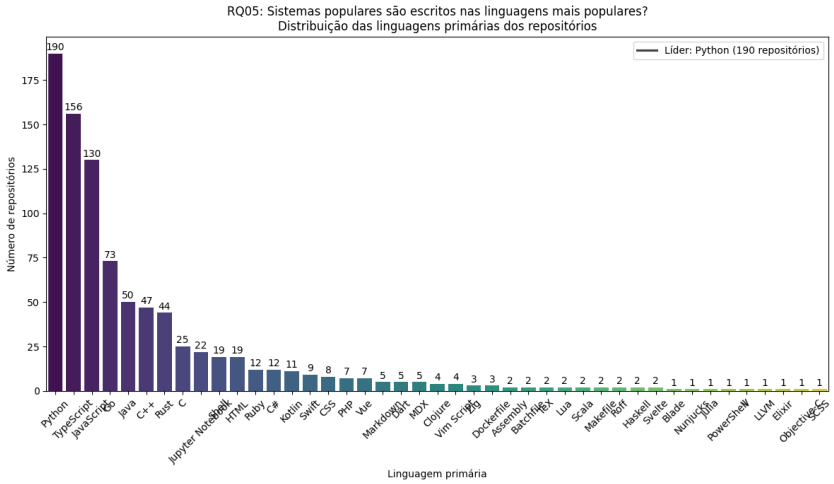
O tempo mediano desde a última atualização é baixo para praticamente todas as linguagens: variando de 0 horas (PHP e C) até 4 horas (Shell). Esses valores confirmam H_a, indicando que os repositórios populares são mantidos de forma ativa e contínua.



Histograma do tempo desde última atualização

RQ05 – Linguagens

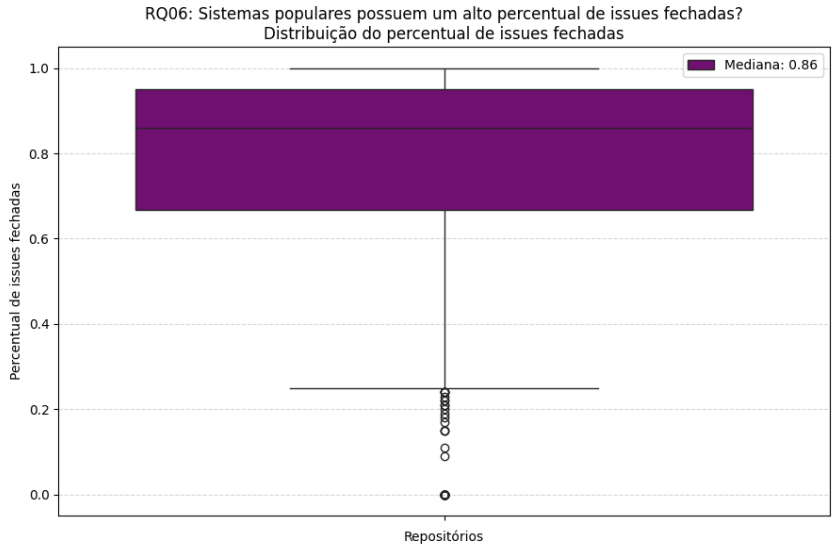
A distribuição mostra que Python (187), TypeScript (156) e JavaScript (130) concentram a maior parte dos repositórios, confirmando H_a de que os sistemas populares se concentram em linguagens mais utilizadas.



Distribuição das linguagens primárias (barplot)

RQ06 – Issues fechadas

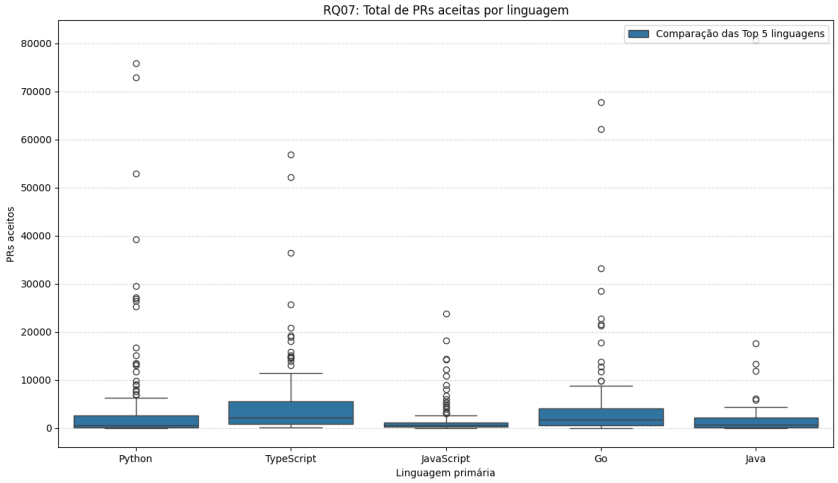
O percentual de issues fechadas foi elevado na maioria dos casos analisados, confirmando H_a de que sistemas populares tendem a apresentar alto índice de resolução de issues.



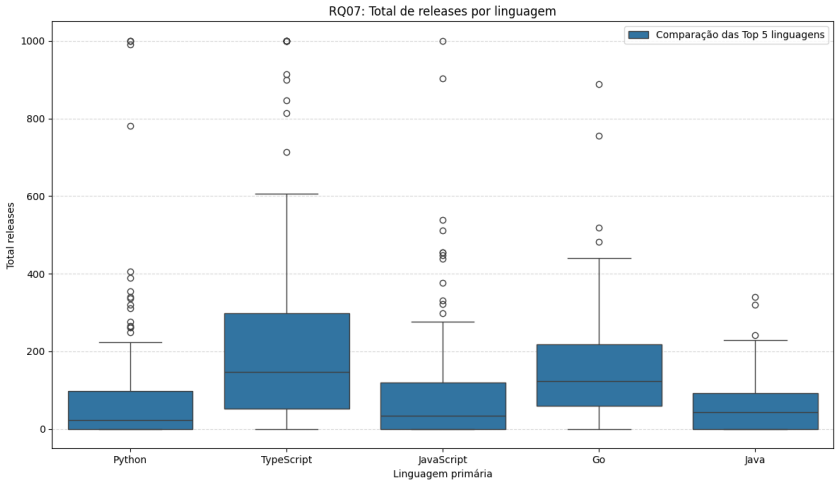
Percentual de issues fechadas (boxplot)

RQ07 – Linguagens e atividade

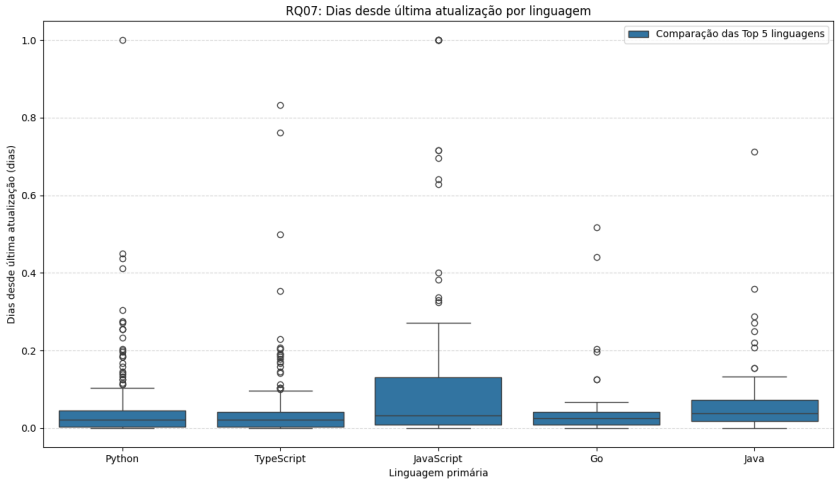
Ao cruzar PRs, releases e tempo de atualização por linguagem, observa-se que linguagens como PHP (2944 PRs, 346 releases, 0 horas desde última atualização), TypeScript (2103,5 PRs, 146,5 releases, 1 hora) e C# (2780,5 PRs, 100 releases, 0,5 hora) apresentam intensa atividade. Por outro lado, Shell (437 PRs, 2 releases, 4 horas) e C (113 PRs, 32 releases, 0 horas) apresentam valores medianos bem menores. Portanto, H_a é parcialmente confirmado: linguagens populares tendem a concentrar mais atividade, mas com exceções relevantes.



Total de PRs aceitos por linguagem (mediana)



Total de releases por linguagem (mediana)



Tempo desde última atualização (dias) por linguagem (mediana)

Considerações

Os resultados confirmam parcialmente as hipóteses: linguagens populares concentram projetos com mais atividade e manutenção. Entretanto, há variação significativa entre as linguagens populares (ex: Shell e C têm valores medianos bem menores). A amostra pode ser enviesada por projetos muito grandes em certas linguagens (ex: PHP com poucos, mas enormes projetos).O tempo desde a última atualização é baixo para quase todos, pois os repositórios populares tendem a ser muito ativos.

6 Conclusão

O estudo das principais características de sistemas open-source populares no GitHub revela um padrão de maturidade, colaboração intensa, entrega contínua e manutenção eficiente. A escolha da linguagem é um fator determinante para o engajamento e sustentabilidade dos projetos, reforçando a importância de ecossistemas ativos para o sucesso do software livre.

Referências

- GitHub Octoverse 2024. Disponível em: `https://github.blog/news-insights/octoverse/octoverse-2024`