PUC Minas
Instituto de Ciências Exatas
e Informática

Pontifícia Universidade Católica de Minas Gerais
Departamento de Engenharia de Software e Sistemas de Informação

Curso de Engenharia de Software
Disciplina: Laboratório de Experimentação de Software
Professor: José Laerte Pires Xavier Junior

Relatório Final - Laboratório 01

Alunos: Guilherme Gabriel Silva Pereira e Lucas Ângelo Oliveira Martins Rocha.

1. Introdução

Neste trabalho está sendo estudado características de sistemas populares open-source, com base nos 1.000 repositórios com maior número de estrelas no GitHub.

A partir disto, algumas hipóteses iniciais para com relação ao estudo foram levantadas. São elas, que os sistemas populares possuem em média, alguns anos de idade desde a sua data de criação no GitHub. Supondo uma idade média de 5 anos para os repositórios destes sistemas populares, tempo necessário para que consiga alcançar uma grande popularidade na comunidade.

Sobre se sistemas populares recebem muitas contribuições externas, imagina-se que o número de *pull requests* com status *merged* dos projetos seja relativamente alto, considerando que seriam repositórios mais maduros, com por volta de 200 *pull requests* nesse estado.

Em relação ao número de *release*, supõe-se uma média de 80. Levando em consideração as hipóteses anteriores, repositórios maduros tendem a ter um acumulado um número considerável de *releases*.

Já em relação à frequência que o sistema recebe atualização, é algo que acredita-se que seja alta, sendo diário ou, no máximo, semanal. Isto pois, para que o repositório tenha possua um bom engajamento da comunidade, é necessário que ele esteja ativo. Além disso, várias ações no repositório podem contar como uma atualização, como fechar uma *issue* ou efetivar *merge* de um *pull request*, até mesmo mudar o texto da documentação *Wiki* poderia atualizar o repositório.

A hipótese mais certeira que pode-se dizer, é que os sistemas mais populares são realmente desenvolvidos nas linguagens mais populares (GITHUB, 2021). Uma conexão direta, pela ideia de que, se algo é desenvolvido em uma linguagem popular, ou a linguagem se populariza, consequentemente poderá ser observado que os conteúdos (neste caso, repositórios) desta linguagem também ficarão populares com desenvolvedores apoiando.

Ademais, imagina-se que sistemas populares tendem a ter um percentual maior de *issues* fechadas em relação ao total de *issues*. Essa hipótese foi elaborada levando em consideração a

proposição de que são repositórios maduros, que já receberam muitas contribuições no passado, que fecharam as *issues*, com isso, tendendo a serem mais estáveis.

Por fim, é esperado também a conclusão de que sistemas escritos em linguagens mais populares recebem mais contribuição externa, lançam mais releases e são atualizados com mais frequência.

2. Metodologia

Neste trabalho, foi desenvolvido um script na linguagem de programação Python, na versão 3.10.4. Este script exerce 3 funções principais para a análise de dados deste trabalho, sendo elas: busca massiva de dados de repositório, conversão do formato de dados para o padrão *comma-separated-values* (csv) e manipulação dos dados transformados para formatos gráficos, estatísticos e informativos. Além disso, também foi feita uma planilha no Google Sheets, com a mesma carga de dados, para gerar os gráficos e informações estáticas, no intuito de validar se todas as informações geradas pelo script coincidiram com um as de um *software* profissional de visualização de dados.

Com relação a busca massiva de dados, o script lê informações dos 1.000 repositórios do GitHub com o maior número de estrelas. Os dados buscados em cada repositórios são o seu id, quantidade de estrelas, nome do criador, url, datas de criação e última atualização, total de *release*, data da última *release*, nome da linguagem de programação primária e as quantidade de *pull requests merged*, *issues* e *issues* com o *status* de fechada. Todas estas informações foram selecionadas para que fosse possível obter respostas para as análises das questões hipotéticas. A obtenção dos dados supra descritos foi realizada através da API 4.0 do Github, que segue a arquitetura GraphQL. Para isso, foi necessário fornecer o *personal token* do Github durante a coleta de dados, feita por meio da biblioteca *request* do Python. Para testes da API, foi utilizada a própria ferramenta de exploração da plataforma do GitHub, que fornece uma interface de consulta, com todos os atributos e entidades que podem ser buscados.

Para formatação dos dados retornados por esta API do GitHub, que são originalmente retornados em *JavaScript Object Notation (JSON)*, foram utilizadas principalmente as bibliotecas *json* para receber os dados e a *csv* para formatá-los e salvá-los em um arquivo no padrão csv. Durante esta formatação, é importante frisar que alguns dados já foram manipulados, sendo eles o cálculo de dias desde a criação do repositório pela data de criação (idade do repositório em dias), dias desde a última atualização (quantidade de dias desde a data que o repositório recebeu algum tipo de atualização) e dias desde a última *release* (quantidade de dias desde a data que foi lançada a última *release*).

A partir disso, com os dados no formato csv e algumas informações de dias já calculadas a partir de datas, foi possível manipular os dados em forma de gráficos e extrair informações estatísticas de forma mais simples. Para isso, foi utilizada principalmente as bibliotecas *pandas,* que possibilitou gerar gráficos do tipo boxplot para representação de grandes quantidades de dados numéricos não relacionados, e a *dataframe_image* com *matplotlib.pyplot,* para conversão das informações em imagens. Ademais, outra função foi feita no script Python para permitir delimitar a moda da linguagem de programação primária dentre toda a lista de mil repositórios.

3. Resultados obtidos

Foram adicionadas as visualizações gráficas geradas pelo Google Sheets e pela biblioteca *pandas* do script em Python, para que seja possível visualizações de diferentes formas e perspectivas de cada informação, apenas para os dados que resultaram em visualizações ligeiramente diferentes por cada ferramenta.

De acordo com as Figuras 1 e Figura 2, foi possível extrair algumas informações relacionadas à idade dos repositórios, calculada em dias a partir das suas datas de criação. Dentre elas, pode-se destacar uma média que, se transformada em anos, resulta em ≅7.2 anos, o repositório mais novo, possuindo apenas 44 dias, pouco mais de 1 mês de sua criação, e também um repositório com quase quinze anos de idade (≅14) no topo do boxplot.

Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
Age in Days	44	1850.5	2610	3271.75	5249

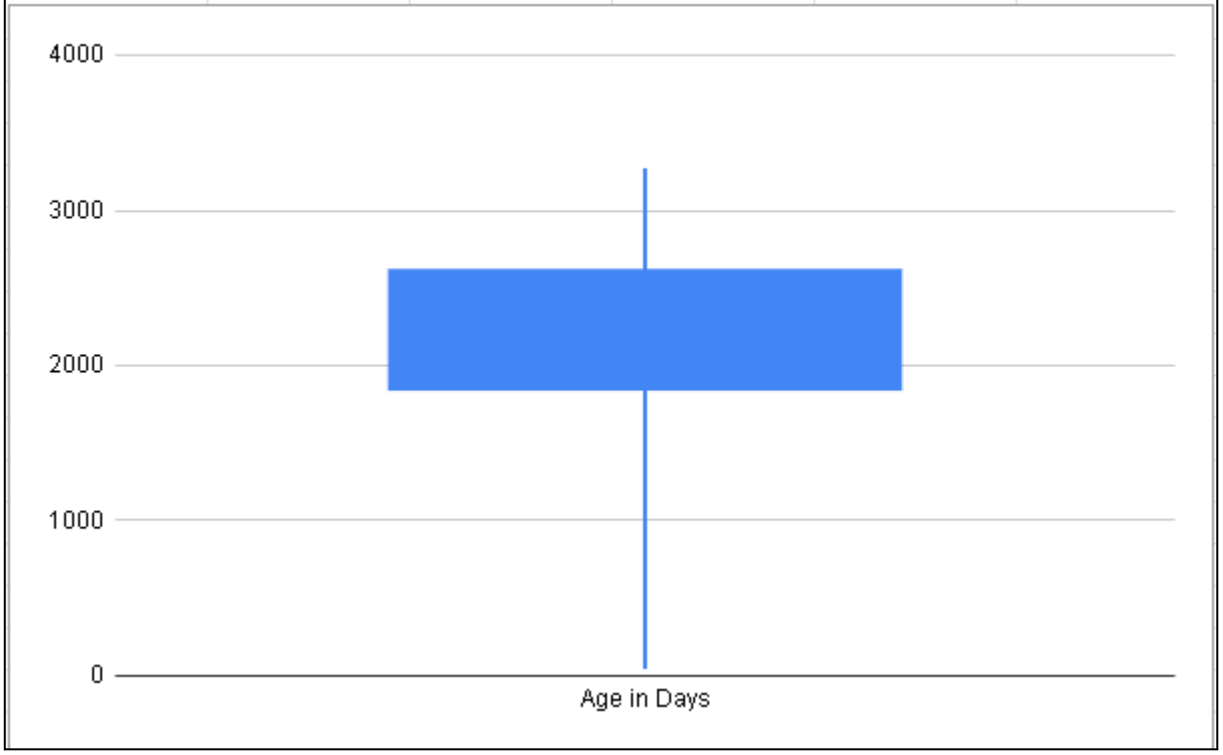

Figura 1 - Gráfico boxplot e informações estatísticas da idade dos repositórios em dias (Google Sheets)

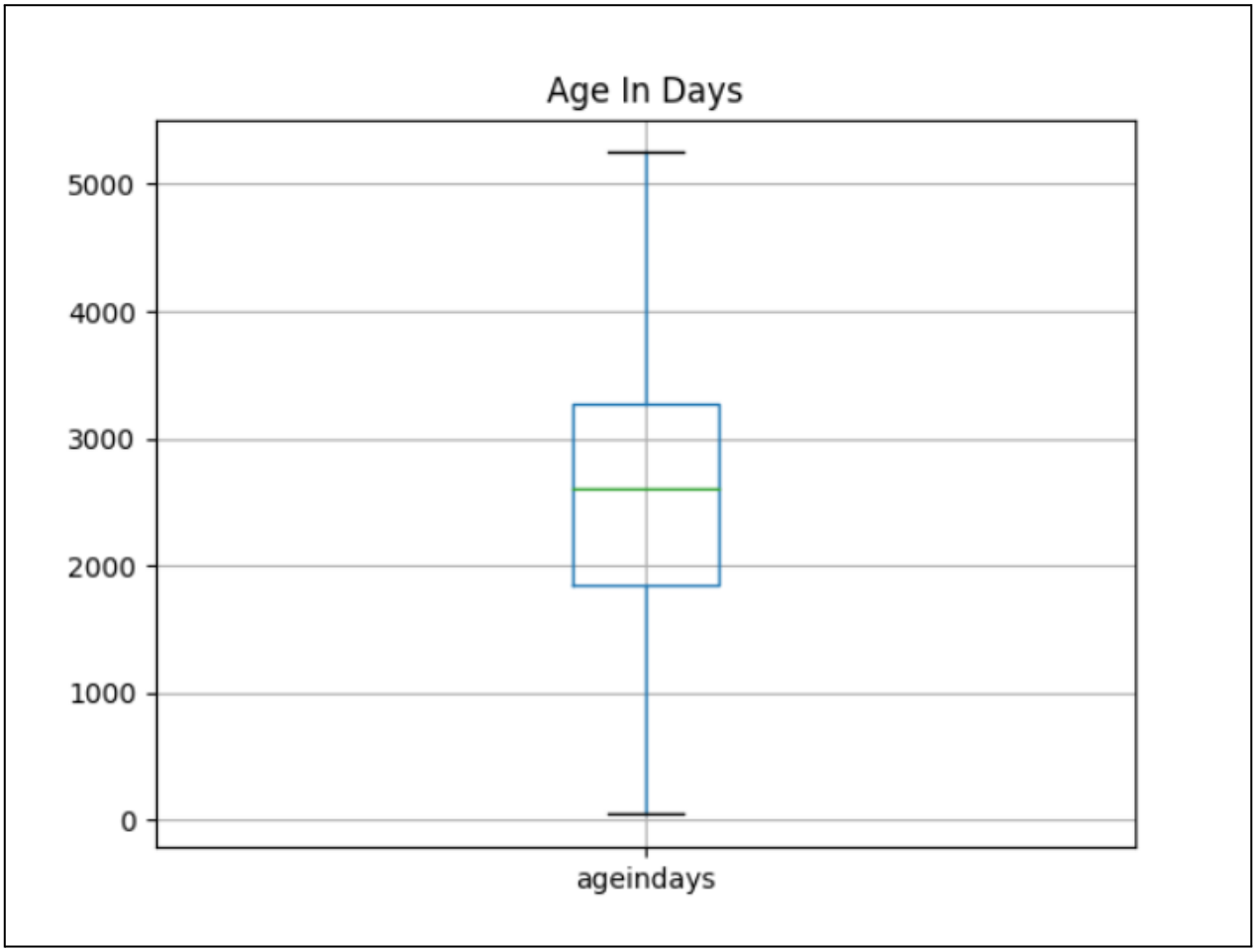

Figura 2 - Gráfico boxplot da idade dos repositórios em dias (Python)

Por meio das Figura 3 e Figura 4, é possível verificar que a grande maioria dos repositórios possuem menos de 500 *pull requests* com *status* de *merged*. Porém, também há repositórios que ultrapassam os milhares de *pull requests* nesse status.

Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
PR merged	0	122	434.5	1484.25	101509

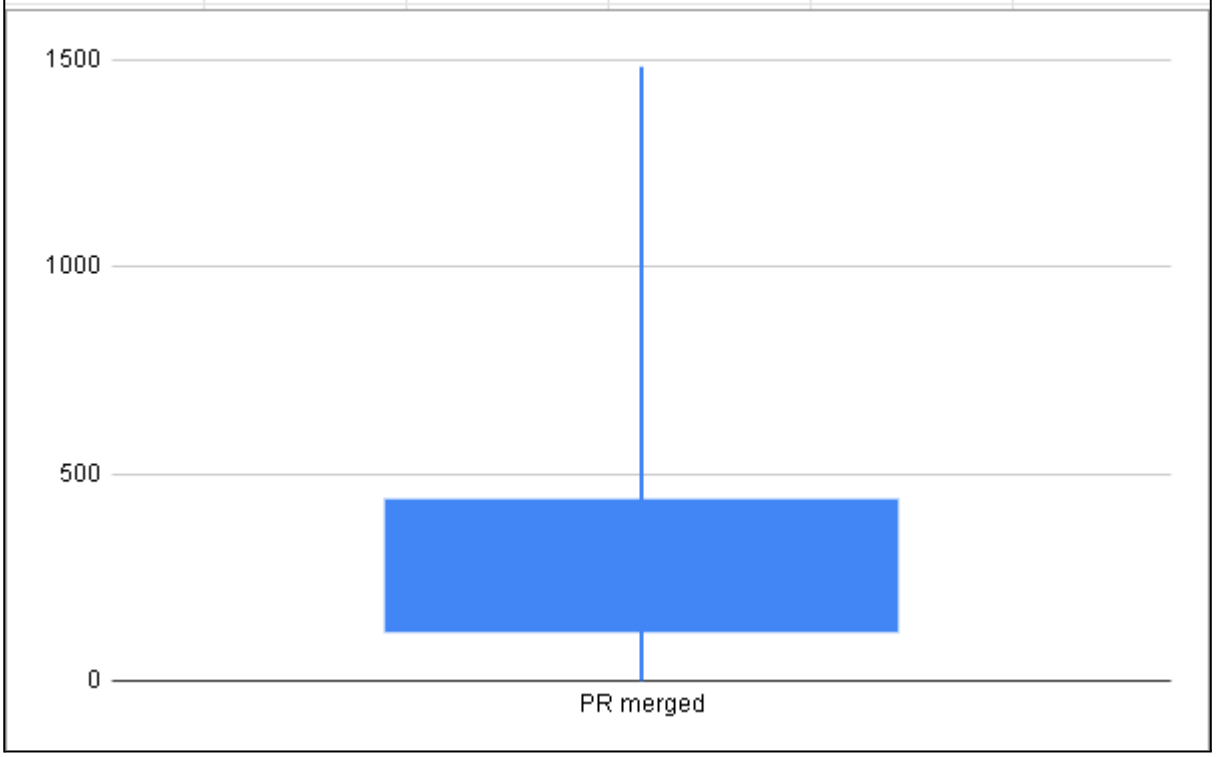

Figura 3 - Gráfico boxplot e informações estatísticas de pull requests com *status* de *merged* (Google Sheets)

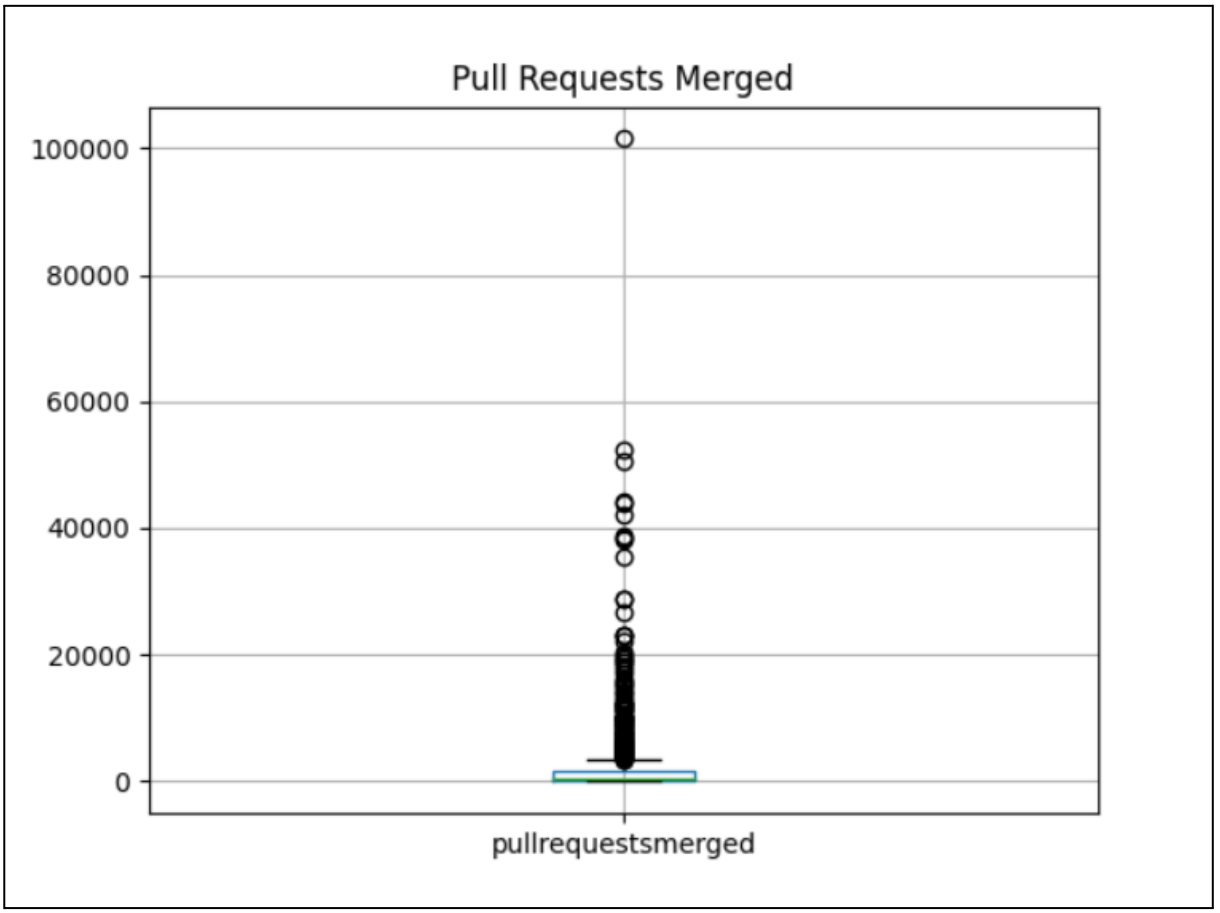

Figura 4 - Gráfico boxplot de pull requests com *status* de *merged* (Python)

A partir das Figura 5 e Figura 6, é passível de visualização a frequência do lançamentos de *releases* dos repositórios tende a 0. Entretanto, possui algumas exceções com quantidade altas de *releases*, notável pelo valor máximo e até mesmo no terceiro quartil.

Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
Releases Count	0	0	19	80	2352

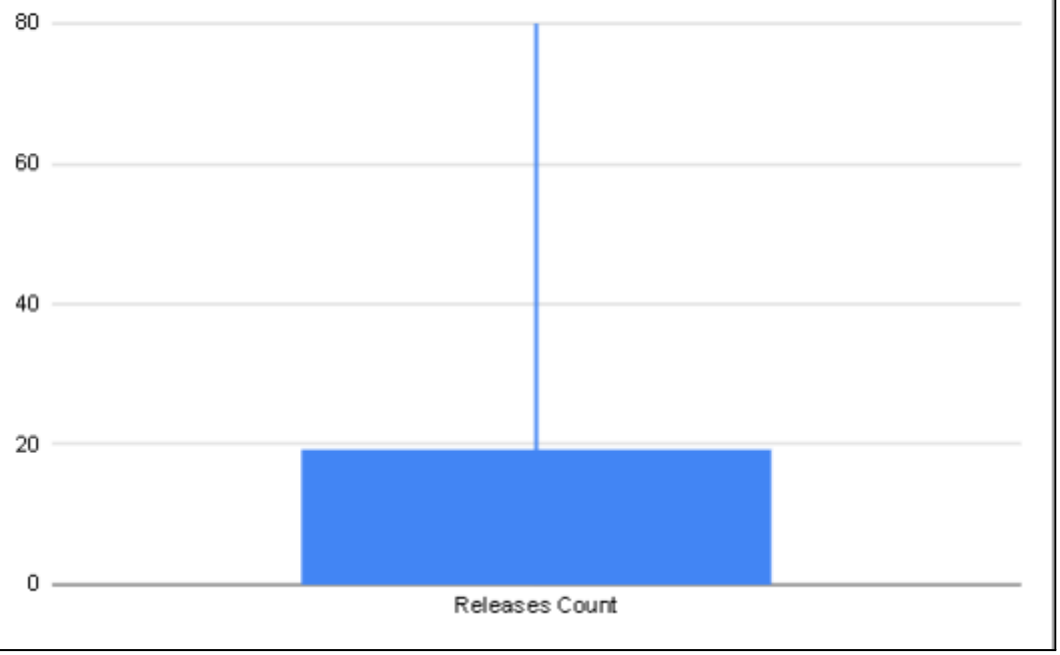

Figura 5 - Gráfico boxplot e informações estatísticas de releases (Google Sheets)

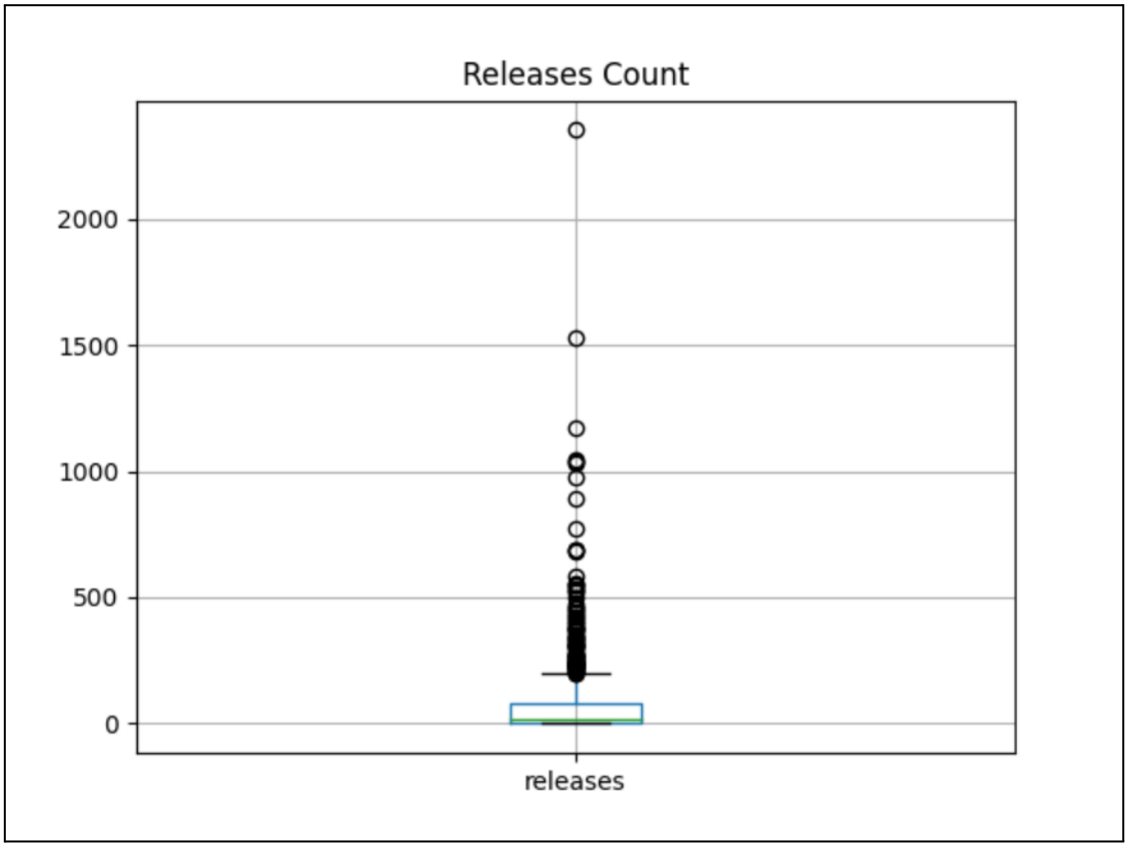

Figura 6 - Gráfico boxplot de releases (Python)

Nas figuras Figura 7 e Figura 8, é perceptível que praticamente a totalidade dos repositórios são atualizados diariamente, isto pois, a quantidade de dias desde a última atualização em ambos os quartis e na média dos repositórios tende a zero. Indicando que não se passa praticamente nem 1 dia sequer sem algum tipo de atualização nestes repositórios, até mesmo o máximo de dias sendo uma quantidade relativamente baixa, três.

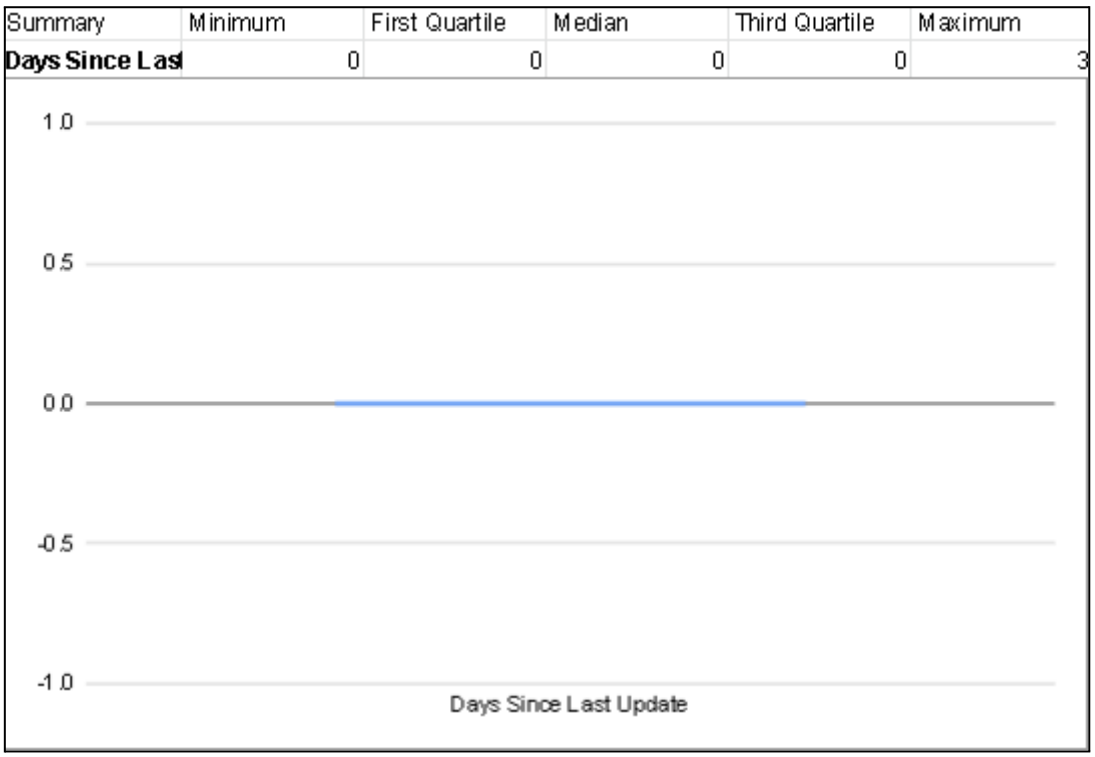

Figura 7 - Gráfico boxplot e informações estatísticas de dias desde a última atualização(Google Sheets)

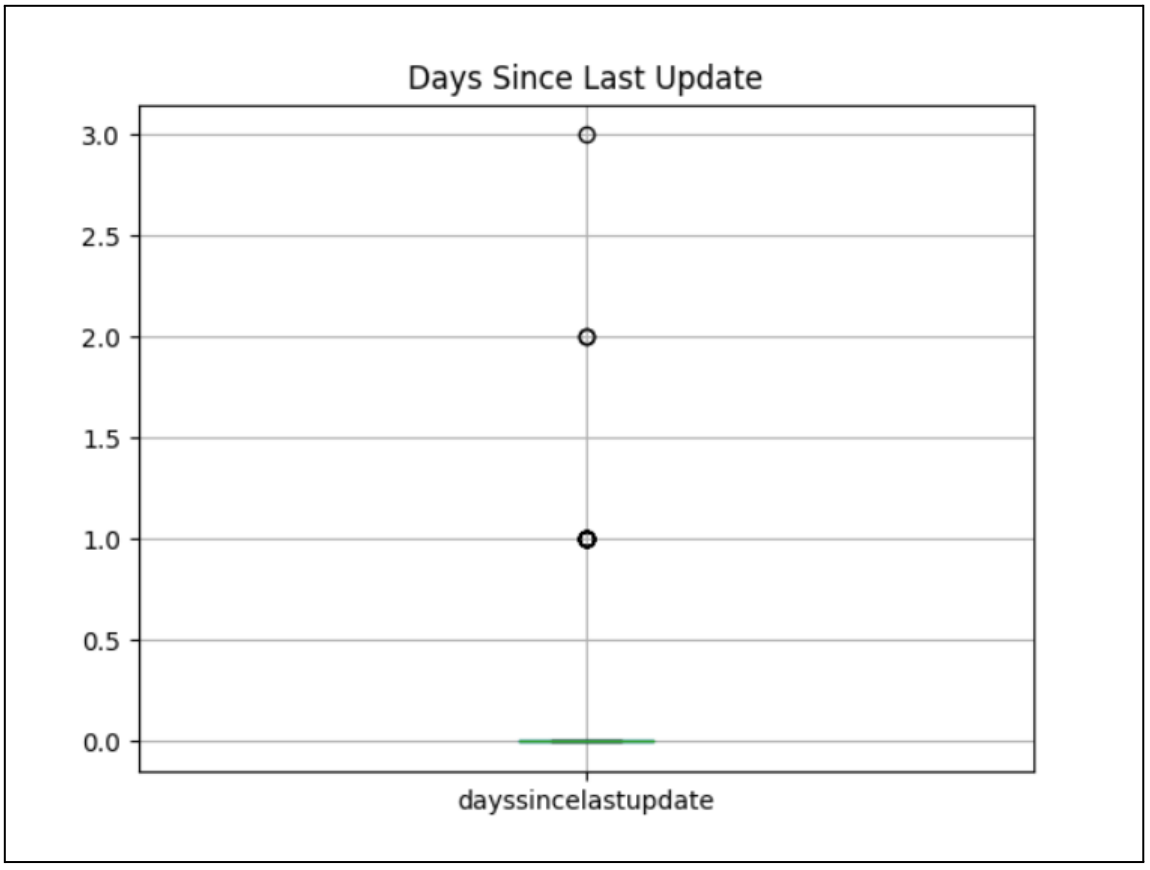

Figura 8 - Gráfico boxplot de dias desde a última atualização (Python)

Em vista das linguagens mais populares dentre todos estes repositórios analisados, a campeã foi JavaScript, com notáveis 227 dos 1.000 repositórios trabalhando primariamente com ela. A

seguir, a Tabela 1, apresenta o ranking das top 10 linguagens primárias mais utilizadas pelos 1.000 repositórios e a respectiva quantidade de repositórios que utiliza de cada uma delas, em ordem decrescente de acordo com a quantia de utilização. É interessante destacar que, dos mil repositórios analisados, 114 não possuem nenhuma linguagem primária.

Nome da Linguagem	**Quantidade de repositório utilizando**
JavaScript	227
Python	112
TypeScript	102
Go	74
Java	69
C++	54
C	29
HTML	23
Shell	22
Rust	20

Tabela 1 - Ranking das top 10 linguagens primárias mais utilizadas pelos 1.000 repositórios com mais estrelas no GitHub (Python)

A partir das Figura 9 e Figura 10, é possível fazer visualizar dados sobre as *issues* dos projetos populares. A distribuição do número de issues se concentra em torno de 250 a 3500, com repositórios alcançando 140.000 issues. Enquanto isso, ao fazer uma relação entre *issues* fechadas com o número total, percebe-se um número alto, com o primeiro quartil sendo de $\cong 75\%$.

Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
Issues Count	0	242.75	1175	3517.25	140348

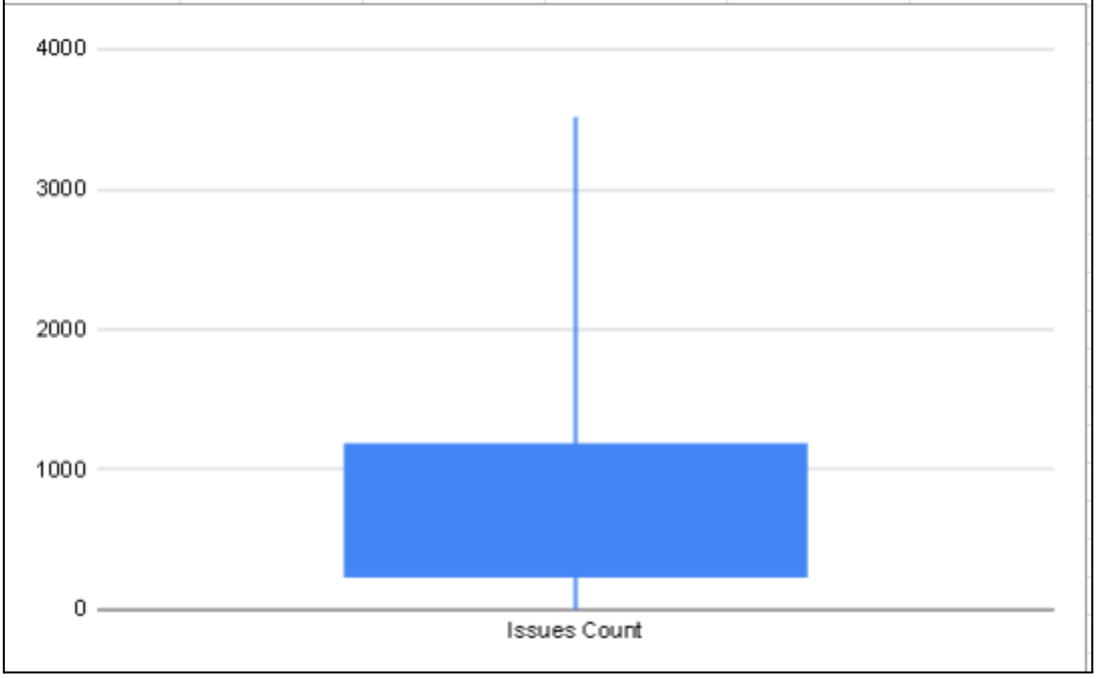

Figura 9 - Gráfico boxplot de issues no total (Google Sheets)

Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
Issues Closed /	0.004296875	0.7154342745	0.8666666667	0.9563310533	1

Figura 10 - Gráfico boxplot de razão de issues fechadas por total de issues (Google Sheets)

Relacionando os dados obtidos por linguagem, foram escolhidas as 10 linguagens mais usadas no Github [1]. A partir da Figura 11 é possível perceber um destaque alto do Ruby, como a linguagem que possui, em média, mais *pull requests* no estado *merged*. Enquanto isso, o destaque

de quantidade de *Releases*, mostrado na Figura 12, é do *TypeScript*. Já na Figura 13, o padrão foi o mesmo do geral, todas linguagens apresentaram 0 dias desde a última atualização.

		PR merged			
Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
JavaScript	1	145.5	453	1145.5	16940
Python	0	90.75	493.5	1824	38563
Java	0	69	350	848	50661
TypeScript	19	534.25	1339.5	2661.25	44157
C#	13	56	389.5	2963.75	11523
PHP	119	449	824	2543	18858
C++	0	230.75	1057.5	5097.75	38848
Shell	16	51	196	519.75	2706
C	0	15	131	1268	12263
Ruby	197	1037	2978.5	9865.75	101509

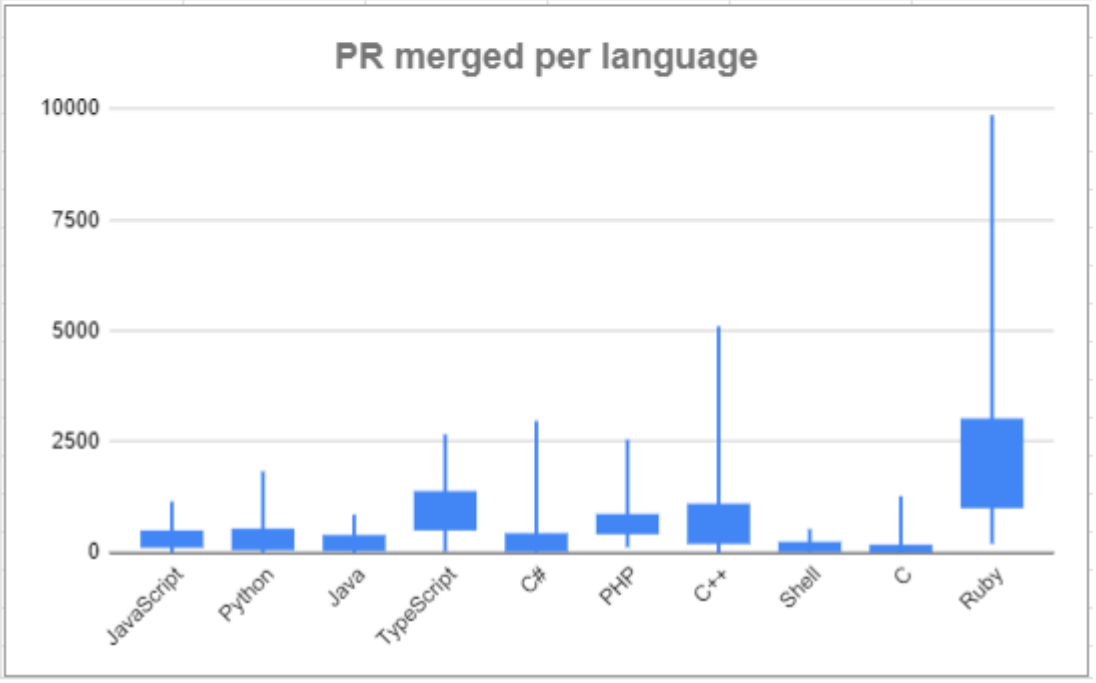

Figura 11 - Gráfico boxplot e informações estatísticas de pull requests com *status* de *merged por linguagem* (Google Sheets)

		Releases Count			
Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
JavaScript	0	0	25	84	1530
Python	0	0	8.5	44.5	975
Java	0	0	22	54	234
TypeScript	0	39.25	99.5	185	2352
C#	1	8.5	60.5	87.25	157
PHP	0	12	56	118	548
C++	0	11	39.5	85.25	1036
Shell	0	0	0	22	227
C	0	0	3	35	160
Ruby	0	0	0.5	80.5	525

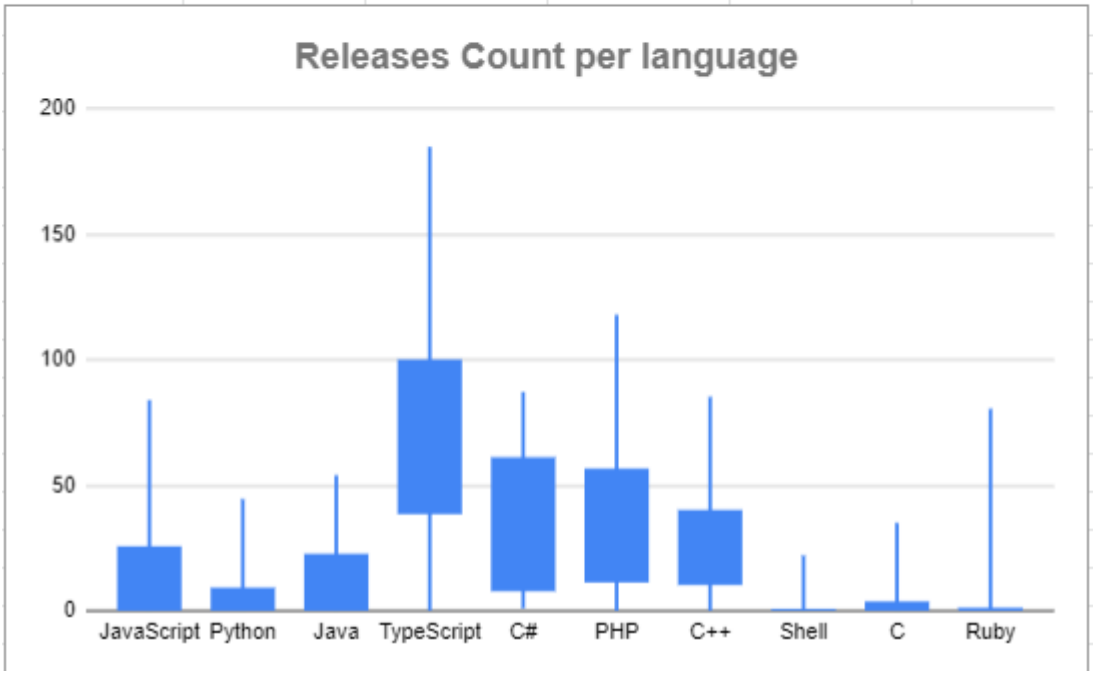

Figura 12 - Gráfico boxplot e informações estatísticas de releases por linguagem (Google Sheets)

		Days Since Last Update			
Summary	Minimum	First Quartile	Median	Third Quartile	Maximum
JavaScript	0	0	0	0	1
Python	0	0	0	0	0
Java	0	0	0	0	1
TypeScript	0	0	0	0	2
C#	0	0	0	0	0
PHP	0	0	0	0	0
C++	0	0	0	0	0
Shell	0	0	0	0	1
C	0	0	0	0	0
Ruby	0	0	0	0	0

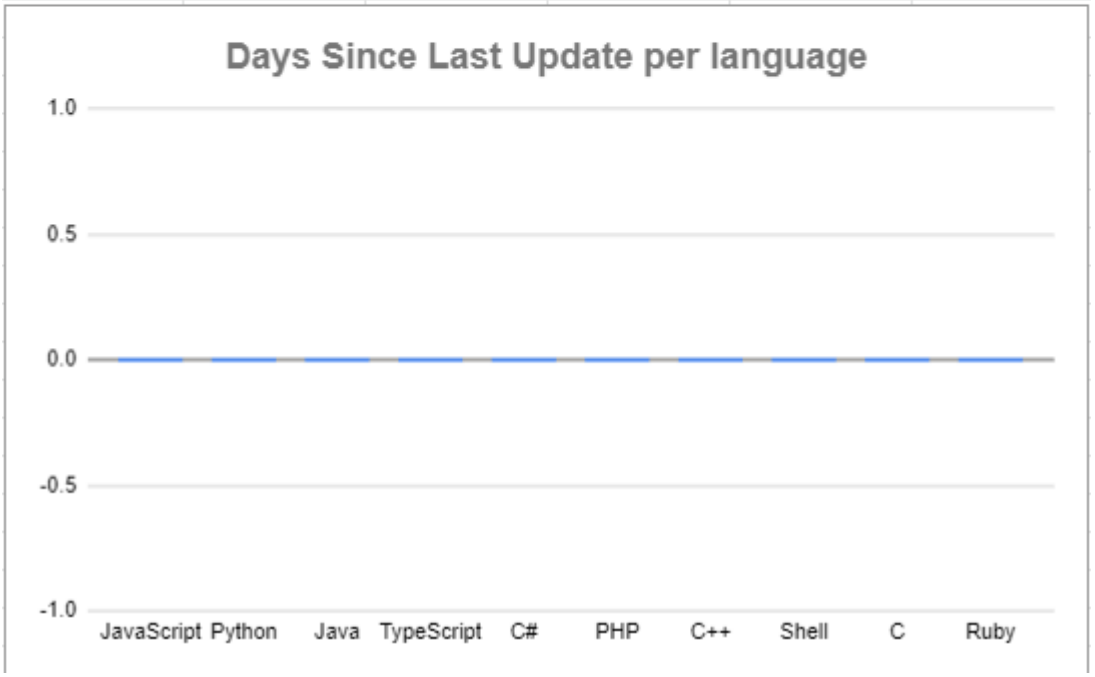

Figura 13 - Gráfico boxplot e informações estatísticas de dias desde a última atualização por linguagem (Google Sheets)

Por fim, na Figura 14, algumas informações estatísticas de todos os dados numéricos analisados.

	stargazerCount	releases	pullrequestsmerged	issues	issuesclosed	ageindays	dayssincelastupdate	dayssincelastrelease
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	659.000000
mean	35620.880000	68.366000	2212.085000	3423.260000	2988.116000	2610.999000	0.022000	327.420334
std	29445.570875	151.855254	6184.229468	7607.481817	6914.632403	1018.287034	0.177616	602.669316
min	18351.000000	0.000000	0.000000	0.000000	0.000000	44.000000	0.000000	0.000000
25%	21332.000000	0.000000	122.000000	242.750000	159.750000	1850.500000	0.000000	9.000000
50%	26092.000000	19.000000	434.500000	1175.000000	966.000000	2610.000000	0.000000	49.000000
75%	37227.500000	80.000000	1485.250000	3517.250000	3116.250000	3271.750000	0.000000	276.000000
max	352034.000000	2352.000000	101510.000000	140375.000000	133214.000000	5249.000000	3.000000	3010.000000

Figura 14 - Informações estatísticas (Python)

4. Expectativas de resultados versus obtidos

A partir dos resultados obtidos, é possível fazer uma análise destes dados, comparando com as hipóteses iniciais, previstas no início deste artigo.

Em relação à idade dos repositórios, a hipótese inicial foi próxima aos resultados obtidos, com uma distribuição principal entre 5 e 9 anos. Conclui-se que os repositórios com mais estrelas no Github são, no geral, maduros em relação a idade.

Sobre contribuições externas em sistemas populares, a distribuição de *pull requests* com status *merged* foi um pouco acima do esperado. O valor imaginado (200) ficou entre o primeiro e o segundo quartil, indicando que sistemas populares possuem, no geral, muitas contribuições externas.

Quando se trata da frequência de *releases*, o valor suposto para quantidade total de *releases* lançados foi bem superior ao obtido, ficando no limiar do terceiro quartil (80). Isso foi contra a hipótese inicial, mostrando que sistemas populares não possuem um alto número de *releases.*

Já no quesito frequência de atualização, o resultado foi de acordo com o esperado, apresentando média de atualização de 0 dias, ou seja, atualização diária. Entretanto, essa métrica se demonstrou um pouco falha para medir a frequência de atualização, pois qualquer ação no repositório altera a data de atualização. Isto é demonstrado ao ver que os dados não se distribuem, com a esmagadora maioria com 0 dias, com um máximo de 3.

Analisando os dados, foi possível perceber também que, sim, os repositórios mais populares são desenvolvidos nas linguagens mais populares. Isso é concluído ao comparar a lista das 10 linguagens mais populares (GITHUB, 2021), com a lista das 10 linguagens que mais apareceram nos dados obtidos. As únicas exceções foram as linguagens Go, HTML e Rust, que apareceram mais que as linguagens populares C#, PHP e Rust.

Outro resultado que se aproximou do esperado, foi a relação entre *issues* abertas e o número total. Foi possível concluir que os repositórios populares costumam ter uma alta porcentagem de *issues* fechadas. Isso pode ser explicado pelo fato de que repositórios populares costumam ter maior engajamento da comunidade em resolver os problemas levantados, além da estabilidade desses repositórios.

Por fim, ao analisar os dados de contribuição externa, *releases* e atualização dos repositórios classificados por linguagem, foi possível perceber que esses dados não são diretamente proporcionais à popularidade da linguagem. Isto é concluído ao analisar que as linguagens do top 3 de popularidade (JavaScript, Python e Java), tem uma distribuição menor que de algumas linguagens que se encontram no meio do top 10, como TypeScript, C# e PHP, além de um destaque para o Ruby, que está na décima posição da lista, mas tem uma média de *pull requests* no estado *merged* exorbitantemente maior que as demais linguagens.

Referências

[1] GITHUB. [Octoverse (2021)]. The State of the Octoverse | The State of the Octoverse explores a year of change with new deep dives into writing code faster, creating documentation and how we build sustainable communities on GitHub. [2021]. Disponível em: https://octoverse.github.com/. Acesso em: 18 ago. 2022.