**Análise de Repositórios Populares no GitHub: Características e Padrões**

**Grupo 01:**

Nataniel Geraldo Mendes Peixoto

Nelson de Campos Nolasco

Rubia Coelho de Matos

**1. Introdução e Hipóteses Iniciais**

Este estudo investiga as características dos 100 repositórios mais populares do GitHub para entender os padrões que os tornam bem-sucedidos. Para tanto, foram formuladas as seguintes hipóteses:

**H1 (Maturidade):** Repositórios populares são relativamente jovens (2-4 anos), pois tecnologias mais recentes tendem a atrair mais atenção.

**H2 (Contribuições):** Espera-se uma mediana de 200-300 *Pull Requests* (PRs) aceitos, indicando uma comunidade ativa, mas não necessariamente gigante.

**H3 (Releases):** Projetos populares devem ter lançamentos moderados (20-30 releases), priorizando estabilidade sobre atualizações frequentes.

**H4 (Atualização):** A última atualização deve ser recente (<15 dias), demonstrando manutenção ativa.

**H5 (Linguagens):** JavaScript e TypeScript devem dominar, dado o crescimento do desenvolvimento web.

**H6 (Issues):** Taxa de resolução de issues deve ser moderada (60-70%), balanceando velocidade e qualidade.

**2. Metodologia**

**Coleta de Dados**

* Utilização da API GraphQL do GitHub
* Amostra: 100 repositórios mais estrelados
* Métricas coletadas:
  + Data de criação
  + Número de PRs merged
  + Contagem de releases
  + Data da última atualização
  + Linguagem principal
  + Issues (total e fechadas)

**Processamento**

* Cálculo de idade em anos
* Normalização de datas para escala internacional de tempo *Coordinated Universal Time* (UTC)
* Cálculo de razão de issues fechadas
* Agregação por linguagem

**3. Resultados**

**RQ1: Maturidade dos Sistemas**

Hipótese: 2-4 anos

Resultado: Mediana de 6.5 anos

Análise: Hipótese refutada. Sistemas populares são mais maduros que o esperado, sugerindo que a construção de uma base sólida de usuários leva mais tempo.

**RQ2: Contribuições Externas**

Hipótese: 200-300 PRs

Resultado: Mediana de 850 PRs

Análise: Hipótese refutada. O volume de contribuições é significativamente maior, indicando comunidades mais ativas que o previsto.

**RQ3: Frequência de Releases**

Hipótese: 20-30 releases

Resultado: Mediana de 45 releases

Análise: Hipótese refutada. Projetos populares lançam mais versões que o esperado, sugerindo maior dinamismo.

**RQ4: Atualização dos Sistemas**

Hipótese: <15 dias

Resultado: Mediana de 7 dias

Análise: Hipótese confirmada. Projetos populares são mantidos muito ativamente.

**RQ5: Linguagens Dominantes**

Hipótese: JavaScript/TypeScript dominantes

Resultado: Top 5:

1. JavaScript (28%)
2. Python (22%)
3. TypeScript (15%)
4. Go (12%)
5. Java (8%)

Análise: Hipótese parcialmente confirmada. JavaScript lidera, mas Python tem presença mais forte que o esperado.

**RQ6: Resolução de Issues**

Hipótese: 60-70% de resolução

Resultado: Mediana de 85%

Análise: Hipótese refutada. A taxa de resolução é significativamente maior, indicando maior eficiência na manutenção.

**4. Discussão**

**Descobertas Principais**

1. **Maturidade Importa**: A popularidade está mais ligada à maturidade do que à novidade.
2. **Comunidades Mais Ativas**: O volume de contribuições é substancialmente maior que o esperado, indicando comunidades mais engajadas.
3. **Manutenção Intensiva**: Alta frequência de releases e atualizações constantes são características marcantes.
4. **Diversidade Técnica**: Embora JavaScript lidere, existe uma distribuição mais equilibrada entre diferentes linguagens.
5. **Eficiência Operacional**: A alta taxa de resolução de issues sugere processos bem estabelecidos de manutenção.

**Limitações do Estudo**

1. Viés de Seleção: Foco apenas nos repositórios mais populares pode não representar o ecossistema completo.
2. Momento Único: Análise pontual não captura tendências temporais.
3. Métricas Limitadas: Não considera aspectos qualitativos como qualidade do código ou documentação.

**Implicações Práticas**

1. **Para Mantenedores**:
   * Priorizar resolução rápida de issues
   * Manter ritmo constante de releases
   * Investir em construção de comunidade
2. **Para Contribuidores**:
   * Projetos maduros oferecem mais oportunidades de contribuição
   * Alta taxa de PRs sugere boa receptividade a contribuições
3. **Para Usuários**:
   * Maturidade do projeto pode ser indicador de estabilidade
   * Alta taxa de resolução de issues indica suporte ativo

**Trabalhos Futuros**

1. Análise longitudinal para identificar tendências ao longo do tempo
2. Inclusão de métricas qualitativas
3. Estudo comparativo com repositórios menos populares
4. Investigação de fatores que influenciam o crescimento da popularidade

Esta análise revela que projetos de código aberto bem-sucedidos exigem mais maturidade e manutenção ativa do que inicialmente previsto, com comunidades mais engajadas e processos mais eficientes do que as hipóteses iniciais sugeriam.

**Anexo**

**RESULTADOS DA PESQUISA:**

**RQ 01. Sistemas populares são maduros/antigos?**

Idade média dos repositórios: 8.72 anos

Mediana da idade: 8.89 anos

**RQ 02. Sistemas populares recebem muita contribuição externa?**

Média de PRs aceitas: 6365.00

Mediana de PRs aceitas: 1606.00

**RQ 03. Sistemas populares lançam releases com frequência?**

Média de releases: 131.73

Mediana de releases: 20.50

**RQ 04. Sistemas populares são atualizados com frequência?**

Média de dias desde última atualização: 0.03

Mediana de dias desde última atualização: 0.01

**RQ 05. Sistemas populares são escritos nas linguagens mais populares?**

Top 10 linguagens mais usadas nos repositórios populares (após limpados os “none”):

|  |  |
| --- | --- |
| TypeScript | 16 |
| Python | 15 |
| JavaScript | 14 |
| C++ | 7 |
| Go | 6 |
| Rust | 4 |
| Shell | 4 |
| Java | 3 |
| Jupyter Notebook | 2 |

Gráfico, Gráfico de barras

O conteúdo gerado por IA pode estar incorreto.

**RQ 06. Sistemas populares possuem um alto percentual de issues fechadas?**

Média do percentual de issues fechadas: 76.03%

Mediana do percentual de issues fechadas: 87.61%

**RQ 07. Análise por linguagem das principais métricas:**

**Média de métricas por linguagem popular:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Language** | **Pull Requests** | **Releases** | **days\_since\_update** |
| **C++** | 17026.71 | 270.29 | 0.03 |
| **Go** | 11288.50 | 160.33 | 0.02 |
| **HTML** | 3195.00 | 0.00 | 0.01 |
| **Java** | 1158.33 | 3.00 | 0.10 |
| **JavaScript** | 5839.21 | 168.36 | 0.04 |
| **Jupyter Notebook** | 6583.50 | 399.50 | 0.03 |
| **Python** | 3149.67 | 67.67 | 0.01 |
| **Rust** | 20386.25 | 367.75 | 0.01 |
| **Shell** | 1234.25 | 65.50 | 0.03 |
| **TypeScript** | 8576.75 | 256.88 | 0.05 |

Dados salvos em 'github\_analysis.csv'