UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RONALDO RUBENS GESSE JÚNIOR

MINERAÇÃO DE REPOSITÓRIOS PARA ANÁLISE DE CICLOS DE SOFTWARE

BAURU
Novembro/2023

RONALDO RUBENS GESSE JÚNIOR

MINERAÇÃO DE REPOSITÓRIOS PARA ANÁLISE DE CICLOS DE SOFTWARE

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Higor Amario de Souza

BAURU
Novembro/2023

Ronaldo Rubens Gesse Júnior

MINERAÇÃO DE REPOSITÓRIOS PARA ANÁLISE DE CICLOS DE SOFTWARE

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Higor Amario de Souza
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profa. Me. Juliana da Costa Feitosa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, _____ de _____ de _____.

Agradecimentos

Agradeço, primeiramente, à Deus por ter me abençoado e permitido estar aqui hoje para realizar este trabalho, um dia tão sonhado. Por ter me dado forças para continuar mesmo nos momentos em que eu achava que eram impossíveis. Agradeço à toda a minha família, especialmente ao meu pai, Ronaldo, à minha mãe, Cátia, ao meu irmão, André, e à minha namorada, Ludmila, que sempre estiveram ao meu lado com todo o apoio para continuar melhorando e crescendo como ser humano, mesmo às vezes com a difícil distância que nos separava. Foram exemplos em minha vida e fizeram o máximo para que eu pudesse estar aqui hoje. Por fim, mas não menos importantes, agradeço aos meus amigos Arthur, Danilo, João Pedro, Nathan e Renato, que fizeram grande parte dessa história, sendo as melhores pessoas que eu poderia conhecer nessa jornada acadêmica.

Até aqui nos ajudou o Senhor!
1 Samuel 7:12

Resumo

Em um cenário tecnológico em constante evolução, a análise para escolha dos componentes e tecnologias de forma assertiva desempenha um papel crucial no sucesso de qualquer projeto de software. *Frameworks* e bibliotecas são componentes essenciais que oferecem funcionalidades ao código e agilizam o processo de desenvolvimento, auxiliando times a entregar um resultado de forma mais eficiente ao usuário final. A mineração de repositórios surgiu como uma forma valiosa de obter informações sobre os códigos-fonte desses softwares, possibilitando análises que visualizem seus ciclos de vida e entendam seu estado atual. Neste trabalho foram analisados 85 softwares com métricas de tendências e correlações para os seguintes dados: número de *commits* e autores dos seus respectivos repositórios de código e nível do interesse relativo, com base nos dados do Google Trends para medir engajamento dos desenvolvedores. A partir disso foi possível analisar os resultados e entender se determinado projeto é realmente viável em seu estado atual de vida, se ele ainda é constantemente atualizado, tem as manutenções devidas e é procurado pela comunidade. Portanto, a aplicação da mineração de repositórios representa um passo importante em direção a um desenvolvimento mais informado e eficaz, alinhado com as necessidades do mercado.

Palavras-chave: Mineração de repositórios, *frameworks*, bibliotecas, ciência de dados, tendência, correlação, análise.

Abstract

In a constantly evolving technological landscape, the analysis for choosing components and technologies accurately plays a crucial role in the success of any software project. Frameworks and libraries are essential components that provide functionality to the code and expedite the development process, assisting teams in delivering results more efficiently to the end user. From this, repository mining has emerged as a valuable way to obtain information about the source code of these software, enabling analyses that visualize their life cycles and understand their current state. In this work, 85 software projects were analyzed with trend metrics and correlations for the following data: the number of commits and authors in their respective code repositories, and the level of relative interest, based on Google Trends data to measure developer engagement. This allowed us to analyze the results and understand whether a particular project is truly viable in its current state of life, whether it is still properly updated, maintained, and sought after by the community. Therefore, the application of repository mining represents an important step towards more informed and effective development, aligned with market needs.
Keywords: Repository mining, frameworks, libraries, data science, trend, correlation, analysis.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

MSR	Mineração de Repositórios de Software
MMS	Média móvel simples
MME	Média móvel exponencial

Sumário

1 Introdução

A área de engenharia de software tem se beneficiado cada vez mais da mineração de dados para extrair informações úteis de diversas fontes, afim de compreender melhor a construção de softwares, identificar tendências e padrões, bem como encontrar problemas e oportunidades para aprimorar a qualidade e a eficiência no processo de desenvolvimento (HASSAN, 2008).

Frameworks e bibliotecas são padrões frequentemente usados na construção de projetos e desempenham um papel importante na arquitetura de sistemas de software complexos (MALDONADO et al., 2002). Ao longo do tempo, esses softwares podem passar por várias etapas de evolução, chamadas de ciclos de vida de softwares, que inclui seu desenvolvimento inicial, manutenção, melhoria e, eventualmente, o declínio.

Ao avaliar os estágios de vida dos softwares, é crucial considerar sua relevância e robustez ao longo do tempo. *Frameworks* e bibliotecas bem estabelecidos normalmente possuem uma comunidade ativa de desenvolvedores, o que é um indicativo de sua sustentabilidade a longo prazo. Além disso, é importante verificar se o suporte e as atualizações de códigos-fonte são frequentes, pois isso indica a saúde e a vitalidade da ferramenta.

A análise apresentada neste trabalho se baseia em dados que provém de duas fontes principais, os repositórios de código para a obtenção de informações sobre a quantidade de *commits* e de autores e a quantidade de pesquisas no Google por projeto para identificar o engajamento da comunidade. Serão investigados os ciclos de vida de *frameworks* e bibliotecas com a intenção de entender quando esses projetos estão em alta ou baixa se baseando nas tendências e correlações sobre a quantidade dos dados. Isso pode fornecer *insights* valiosos para desenvolvedores, gerentes de projeto e pesquisadores que utilizam essas ferramentas, decidindo pela adoção ou substituição de determinados softwares

Ao término, este trabalho disponibiliza duas entregas, sendo a primeira uma análise das métricas de tendências e correlações dos projetos escolhidos e a segunda um código-fonte baseado na metodologia proposta, que possibilita o usuário interagir e criar suas próprias análises conforme sua necessidade.

2 Fundamentação Teórica

Nesta seção serão introduzidos conceitos base sobre *frameworks* e bibliotecas, repositórios de código, mineração de repositório, análise de tendência e correlações de dados. Estes fundamentos serão importantes para a compreensão do projeto apresentado.

2.1 *Frameworks* e bibliotecas

Frameworks e bibliotecas são componentes utilizados no desenvolvimento de software. Eles oferecem um conjunto de funcionalidades pré-desenvolvidas que os programadores podem usar para facilitar e acelerar o processo de desenvolvimento.

Um *framework* é uma estrutura que fornece uma arquitetura organizada para o desenvolvimento de uma aplicação (DUARTE, 2015). Os *frameworks* podem incluir uma variedade de componentes, como gerenciamento de rotas, autenticação de usuários, manipulação de banco de dados e muito mais. Eles são projetados para facilitar a vida dos desenvolvedores, permitindo que se concentrem em aspectos mais específicos do desenvolvimento, em vez de recriar funcionalidades comuns.

Uma biblioteca é um conjunto de funções e rotinas que podem ser utilizadas para realizar tarefas específicas (DUARTE, 2015). Ela não possui uma estrutura organizacional rígida como um *framework*, em vez disso, as bibliotecas fornecem funcionalidades que os desenvolvedores podem chamar conforme necessário. Um exemplo clássico é a biblioteca Pandas (Python), que simplifica e adiciona funções de auxílio para a manipulação de dados. Os desenvolvedores podem incorporar bibliotecas em seus projetos e utilizá-las para economizar tempo e esforço.

A principal diferença entre esses softwares está no controle do desenvolvedor. Em um *framework*, o controle é invertido, o que significa que o desenvolvedor escreve código que se encaixa nas estruturas e convenções fornecidas pelo *framework*. Já com uma biblioteca, o desenvolvedor mantém o controle e chama as funções da biblioteca conforme necessário (MALDONADO et al., 2002).

2.2 Controle de versão

O controle de versão de código é um processo que envolve rastrear e gerenciar alterações feitas no código-fonte de um projeto de software ao longo do tempo. Ele permite que várias pessoas colaborem em um projeto de desenvolvimento de software ao mesmo tempo sem gerar conflitos entre as versões do produto.

Os repositórios de código servem como depósitos onde o código-fonte de um projeto é

armazenado, gerenciado e compartilhado. Estes depósitos são fundamentais para o controle de versões e para o paralelismo no desenvolvimento.

2.2.1 Git

O Git é uma ferramenta de controle de versão muito utilizada no meio do desenvolvimento de softwares. Ele permite que equipes de desenvolvedores colaborem de forma eficaz em projetos, mantendo um histórico preciso de todas as modificações realizadas no código. Isso é alcançado através da criação de *commits*, que são um retrato do estado do código em um determinado momento.

Com ele, um projeto pode evoluir de forma mais rápida e controlada, considerando que os programadores tem a liberdade de trabalhar no código ao mesmo tempo sem que haja alguma confusão entre as versões finais.

2.3 Mineração de repositórios de software (MSR)

A MSR é uma técnica utilizada para obter informações úteis a partir dos dados armazenados nos códigos-fonte de softwares. Esta técnica emprega algoritmos e métodos analíticos para acessar e transformar os dados, afim de identificar padrões, comportamentos e *insights* que podem ser relevantes para aprimorar o desenvolvimento de projetos (HASSAN, 2008).

Com isso, a mineração desses dados engloba uma ampla gama de análises, incluindo a identificação de padrões e a evolução do projeto ao longo do tempo. Ela é frequentemente aplicada para prever tendências futuras no desenvolvimento de software, permitindo que equipes antecipem possíveis desafios.

2.4 Medidas de avaliação

2.4.1 Interesse relativo

O interesse relativo se refere à avaliação da popularidade ou relevância de um determinado assunto em relação a um período de tempo ou contexto específico. Esta análise é valiosa para compreender a dinâmica de interesse do público em relação a temas específicos, no caso desse trabalho, de *frameworks* e bibliotecas.

2.4.2 Análise de tendência

A análise de tendência é um estudo científico de padrões e variações que os dados podem indicar ao longo de um período específico. Para identificar e quantificar tendências, os analistas

podem empregar uma variedade de métodos estatísticos e técnicas de modelagem, como por exemplo a média móvel simples, exponencial e a análise de séries temporais. Normalmente essas técnicas são aplicadas para análises financeiras, como na verificação de tendência em preços (ELDER, 2006), sendo possível também utilizar essas métricas em outras áreas, como nos dados que serão aqui analisados.

2.4.3 Média Móvel Simples (MMS)

A MMS é utilizada para suavizar variações em séries temporais de dados, tornando mais simples identificar padrões de longo prazo. Essa média é calculada definindo uma variável "n"como tamanho da janela de observação dos dados "V", somando as informações dos últimos "n" períodos e dividindo pela mesma quantidade.

Segue sua equação:

$$MMS = \frac{V_t + V_{t-1} + ... + V_{t-n}}{n} \tag{2.1}$$

2.4.4 Média Móvel Exponencial (MME)

A MME é utilizada para suavizar variações em séries temporais de dados, proporcionando uma visão mais sensível às mudanças recentes do que às mudanças de longo prazo. Essa média é calculada através de uma fórmula que leva em conta o valor atual, o valor anterior da MME e um fator de ponderação. Esse fator determina a rapidez com que a MME reage a novos dados.

Assim como descrito na Seção 2.4.3, a MME utiliza uma variável "n"para definição do período da janela dos dados "V", além do fator de ponderação "K", calculado como: K = 2/(n+1).

Segue sua equação:

$$MME = V * K + MME_{anterior} * (1 - K) \tag{2.2}$$

2.4.5 Correlações estatísticas

Como descrito em (SOUSA, 2019), as correlações estatísticas são medidas que quantificam relações ou associações entre duas ou mais variáveis em um determinado conjunto de dados. Elas são importantes para analisar como as informações se comportam e se há alguma relação de dependência entre elas.

As duas correlações utilizadas nesse trabalho são a de Pearson e a de Spearman. Ambas produzem um coeficiente de correlação independente da unidade de medida das variáveis que tem resultados possíveis de -1 a $+1$, sendo que $+1$ indica uma correlação positiva perfeita,

enquanto -1 indica uma correlação negativa perfeita. Caso o resultado seja 0, não existe uma relação linear perceptível entre as variáveis.

2.4.5.1 Correlação de Pearson

A correlação de Pearson é uma medida estatística que avalia a relação linear entre duas informações (as variáveis se movem na mesma direção, a uma taxa constante) (SOUSA, 2019). Caso essa correlação seja aplicada em dados não lineares, poderá haver perda na acurácia dos resultados.

2.4.5.2 Correlação de Spearman

A correlação de Spearman, por sua vez, é uma medida de correlação que avalia a relação monotônica entre duas informações (as variáveis tendem a uma mesma direção, mas não necessariamente a uma taxa constante) (SOUSA, 2019). Ela é baseada na ordenação dos valores e não assume uma relação linear entre as variáveis. Essa correlação não é sensível a dados dispersos, ou seja, não exige que os dados analisados tenham distribuições normais.

2.5 Ferramentas

Nesta seção serão introduzidas as ferramentas utilizadas em todo o processo do trabalho, desde a obtenção dos dados, tratamento, visualização completa e análises.

2.5.1 Python

O Python é uma linguagem de programação criada por Guido van Rossum, com sua primeira versão lançada em 1991. Sendo de alto nível e orientada a objeto, fornece uma ampla gama de possibilidades para projetos de forma facilitada e eficaz com uma programação mais intuitiva.

Neste trabalho, optaremos em utilizar a linguagem Python pelas seguintes vantagens:

- Possui uma sintaxe limpa e simples de ler, o que torna o código mais compreensível. Isso facilita a colaboração e a manutenção do projeto.
- É compatível com diversas plataformas, incluindo Windows, macOS e Linux. Isso torna possível que o mesmo código possa ser executado em diferentes sistemas operacionais.
- Possui uma grande quantidade de bibliotecas padrões que abrangem uma ampla gama de funcionalidades. Para esse projeto, focaremos no ferramental voltado para a área de ciência de dados.
- Possui uma comunidade extensa e ativa de desenvolvedores, o que significa que há muitos recursos, documentação e suporte disponíveis em larga escala.

A documentação completa pode ser encontrada em (PYTHON, 2023).

2.5.2 Pandas

A biblioteca Pandas é amplamente utilizada por sua simplicidade e capacidade de manipulação e análise de dados. Essa ferramenta visa ser uma das principais escolhas para profissionais de ciência de dados por fornecer estruturas de dados eficientes, ter facilidade em transformar informações, boas integrações com outras bibliotecas, além de ter uma comunidade e documentação ativas. A documentação completa pode ser encontrada em (PANDAS, 2023).

2.5.3 Matplotlib

O Matplotlib é uma biblioteca utilizada para a visualização e análise de dados. Com ela é possível criar uma variedade de gráficos, como exemplo os gráficos de dispersão, gráficos de barras, gráficos de pizza, gráficos de linhas e histogramas. O Matplotlib é relativamente fácil de aprender e usar, especialmente para usuários que já estão familiarizados com Python. A documentação completa pode ser encontrada em (MATPLOTLIB, 2023).

2.5.4 PyDriller

O PyDriller é um *framework* em Python utilizado para simplificar a obtenção de informações importantes a partir de repositórios Git. Ele permite a extração de dados sobre *commits*, como exemplo o numero modificações em arquivos, autores e data das modificações.

Uma das vantagens marcantes do PyDriller é a sua simplicidade e eficácia no uso. Com uma sintaxe limpa e fácil, os desenvolvedores podem facilmente começar a explorar repositórios sem a necessidade de conhecimento extensivo sobre Git. A documentação completa pode ser encontrada em (PYDRILLER, 2023).

2.5.5 PyTrends

O PyTrends é uma API que oferece uma interface Python para acessar a API do Google Trends de maneira simples, permitindo aos analistas extrair dados sobre o interesse relativo para determinados assuntos de busca ao longo do tempo.

Os usuários podem realizar consultas de tendências de busca, visualizar dados em diferentes intervalos de tempo, em várias localizações geográficas e em diferentes categorias de pesquisa. A documentação completa pode ser encontrada em (PYTRENDS, 2023).

2.5.6 Jupyter

O Jupyter é uma ferramenta que fornece um ambiente totalmente interativo para desenvolvimento de softwares, permitindo a criação e compartilhamento de *notebooks*, documentos

que combinam código executável, visualizações e marcadores em formato de texto.

Com isso é possível aliar a narrativa de construção e execução do projeto com o código-fonte, facilitando a utilização, visualização e replicação do trabalho. A documentação completa pode ser encontrada em (JUPYTER, 2023).

3 Metodologia

Neste capítulo serão apresentados os passos utilizados durante todo o processo do projeto, desde a obtenção dos dados via PyDriller e PyTrends, passando pela transformação dos dados com a biblioteca Pandas, até a visualização final utilizando métodos estatísticos e a biblioteca Matplotlib.

3.1 Seleção de softwares

A escolha dos frameworks e bibliotecas foi dividida em etapas para facilitar e abranger a maior quantidade e qualidade dos dados. Com isso, a seleção do software foi baseada na sua área de atuação, linguagem de programação e período ativo. Primeiro foram selecionadas seis áreas distintas de atuação de cada software: *machine learning*, web, ciência de dados, API Rest, teste e segurança. Como próximo passo foram escolhidas cinco linguagens de programação atuais e consolidadas no mercado, sendo: Python, C++, Java, Ruby e JavaScript.

Como ultima etapa, foram levados em consideração tanto projetos atuais e ainda ativos, quanto aqueles que já são legado, não tem manutenção ativa e são pouco utilizados. Com isso, será possível basear a análise em ciclos de vida já completos no caso dos projetos legado, gerando *insights* valiosos que podem ser aplicados nos softwares ainda em atividade plena. Para essa seleção foram levadas em consideração as seguintes características:

- Para softwares atuais, foram escolhidos aqueles que são muito utilizados em sua determinada área e linguagem, seja pela fama em fóruns e projetos, ou pelas atividades frequentes em seus códigos-fonte.
- Para softwares legado, foram escolhidos aqueles que já estão quase sem utilização em projetos atuais, com atividades recentes quase nulas em seus códigos-fonte

Ao todo foram selecionados dois projetos por área por linguagem no caso dos atuais, e um por área por linguagem no caso dos *frameworks* e bibliotecas legado, resultando oitenta e cinco softwares, sendo sessenta atuais e vinte e cinco legados. Houve uma dificuldade em encontrar alguns projetos legados pois a maioria teve seu link para o repositório original expirado.

Segue uma amostra dos softwares selecionados:

Tabela 1 – Seleção de softwares ativos

Área	**Python:**	**C++:**	**Java:**	**Ruby:**	**JavaScript:**
Machine Learning:	TensorFlow	TensorFlow	Deeplearning4j	TensorFlow.rb	TensorFlow.js
	PyTorch	PyTorch	H2O	SciRuby	Brain.js
Web:	Django	Wt	PlayFramework	Ruby on Rails	React
	Flask	cpprestsdk	Apache Struts	Hanami	Vue.js
Ciência de Dados:	pandas	Armadillo	MOA	NMatrix	D3.js
	Dask	ITK	WEKA	Statsample	Chart.js
API Rest:	FastAPI	Pistache	Spring REST	Grape	Express.js
	Eve	Restbed	RESTEasy	Roda	Koa
Teste:	pytest	Google Test	JUnit5	RSpec	Jest
	Robot Framework	Catch2	TestNG	Minitest	Cypress
Segurança:	OWASP ZAP	OpenSSL	OWASP Java Encoder	Brakeman	Node.js Security
	Paramiko	Botan	Bouncy Castle	Sorcery	helmet

Tabela 2 – Seleção de softwares legados

Área	**Python:**	**C++:**	**Java:**	**Ruby:**	**JavaScript:**
Machine Learning:	Orange3	CNTK	JakartaeeFaces	Ruby FANN	Synaptic.js
Web:	Web2py	CppCMS	Apache Struts	Camping	Backbone.js
Ciência de Dados:	PyTables	mlpack	Apache Mah	Scruffy	js-stat
API Rest:			Apache CX	Sinatra Classic	Restify
Teste:	unittest2	CppUnit	JUnit 4	Test::Unit	QUnit
Segurança:	PyCrypto	Libtom	Java Cryptography Extension (JCE)	Ruby OpenSSL	jsrsasign

3.2 Obtenção dos dados

Com a seleção de *frameworks* e bibliotecas definida na Seção 3.1, foi utilizada a ferramenta Jupyter para criar dois *notebooks* iniciais, um para a MSR com o objetivo de ter informações sobre os *commits* e outro para adquirir informações sobre a quantidade de pesquisas via Google Trends.

3.2.1 *Commits*

A ferramenta Pydriller foi utilizada para obter dados de *commits* dos oitenta e cinco repositórios de código baseando-se em seus respectivos links.

Como a quantidade selecionada de *frameworks* e bibliotecas é extensa, a obtenção de todas essas informações em um único processo levariam aproximadamente 3 a 4 dias de processamento com uma internet de 300 Mbps (megabytes por segundo). Para o sucesso da extração foi necessária a utilização de três outros *notebooks* Jupyter idênticos ao de MSR e de quatro arquivos distintos contendo os links para os repositórios. Dessa forma os códigos foram executados de forma paralela, porém, extraindo dados complementares, diminuindo o tempo total em três vezes. Com essas informações, foi possível gerar gráficos temporais importantes que possibilitam a análise sobre o ciclo de vida de determinado software, como a quantidade de *commits* e autores por tempo.

Como um exemplo de parte prática, a mineração de dados do repositório da biblioteca Pandas utiliza o seguinte código:

```
pip install pydriller
from pydriller import Repository
import pandas as pd
import datetime as dt

# Criação de uma lista vazia.
response = []
# Percorre todos os commits do repositório extraindo as informações.
for commit in Repository(['https://github.com/pandas-dev/pandas']).
traverse_commits():
    print(commit.project_name)
    # Concatena todas as informações na lista "response".
    response.append(f"{commit.author.name},,,,{commit.committer},,,,
    {commit.hash},,,,{commit.committer_date},,,
    {commit.project_name},,,{commit.deletions},,,
    {commit.insertions},,,,{commit.msg}")
# Transforma a lista "response" em um dataframe.
data = pd.DataFrame(response, columns = ["Commits"])
```

Com ele foram obtidas diversas informações dos *commits*, como nome do autor, id, data, nome do projeto, quantidade de linhas deletadas, quantidade de linhas inseridas e a mensagem inserida.

Nesse exemplo, foi passado apenas um link para a extração dos dados, referente ao repositório git da biblioteca Pandas. Para o atual trabalho, foram disponibilizadas em quatro listas os links dos repositórios atuais e legados escolhidos.

A tabela inicial com informação dos *commits* é apresentada em uma única coluna, com todos os dados concatenados e separados por ',,,':

Figura 1 – Exemplo inicial da tabela de *commits*

	Commits
0	Wes McKinney,,,,<pydriller.domain.developer.De...
1	Wes McKinney,,,,<pydriller.domain.developer.De...
2	Wes McKinney,,,,<pydriller.domain.developer.De...
3	Wes McKinney,,,,<pydriller.domain.developer.De...
4	Wes McKinney,,,,<pydriller.domain.developer.De...
...	...
33437	jbrockmendel,,,,<pydriller.domain.developer.De...
33438	jbrockmendel,,,,<pydriller.domain.developer.De...
33439	jbrockmendel,,,,<pydriller.domain.developer.De...
33440	jbrockmendel,,,,<pydriller.domain.developer.De...
33441	Natalia Mokeeva,,,,<pydriller.domain.developer...

33442 rows × 1 columns

Fonte: Elaborada pelo autor.

Na Seção 3.3 esses dados serão separados em suas respectivas colunas e tratados adequadamente para possibilitar uma leitura e um manejo mais simples.

3.2.2 *Trends*

A ferramenta PyTrends foi utilizada para obter informações de pesquisa da API do Google Trends afim de mensurar a popularidade de determinado software ao longo do tempo. A medida resultante é o interesse relativo por mês, o que também colabora para entender ciclos de vida dos softwares com o engajamento da comunidade.

Como um exemplo de parte prática, a extração dos dados sobre interesses relativos da biblioteca Pandas utiliza o seguinte código:

```
from pytrends.request import TrendReq
import pandas as pd

# Listagem de termos para pesquisa.
termos = ["Pandas"]
# Crie uma instância da TrendReq.
```

```
# Defina a linguagem (hl) e o fuso horário (tz).
pytrends = TrendReq(hl='pt-BR', tz=360)
# Configure os parâmetros da busca.
pytrends.build_payload(termos, cat=5, timeframe='all', geo='',
gprop='')
# Obtenha os dados.
trends = pytrends.interest_over_time()
```

Nesse código, há um parâmetro muito importante que tem relação direta com a abrangência e confiabilidade dos dados obtidos. O parâmetro 'cat' é a categoria em que a busca será realizada, nesse caso, 'cat' $= 5$ direciona para o assunto '*Computers and Electronics*' segundo a documentação (PYTRENDS, 2023). Caso esse parâmetro não existisse, a biblioteca *Node* por exemplo teria um número muito maior de interesse relativo do que realmente tem, por ser uma palavra que pode ser facilmente pesquisada em outro contexto.

Já a lista de termos representa o nome dos *frameworks* e bibliotecas utilizados. Para o trabalho, foram utilizadas também variantes dos nomes originais, considerando letras maiúsculas ou minúsculas, espaços e pontuações.

A tabela inicial com informação das *trends* é apresentada em três colunas, sendo a primeira de data, a segunda com o interesse relativo e uma terceira que aponta caso os dados sejam parciais ou totais:

Figura 2 – Exemplo inicial da tabela de *trends*

date	Pandas	isPartial
2004-01-01	0	False
2004-02-01	1	False
2004-03-01	0	False
2004-04-01	0	False
2004-05-01	4	False
...	...	...
2023-06-01	69	False
2023-07-01	64	False
2023-08-01	69	False
2023-09-01	75	False
2023-10-01	62	True

238 rows × 2 columns

Fonte: Elaborada pelo autor.

3.3 Transformação dos dados

Na extração realizada na Etapa 3.2, tanto os dados de *commits* quanto os de *trends* são gerados sem nenhum tratamento, ou seja, as informações vem desorganizadas, não há padrão na nomenclatura das colunas e as datas são do tipo *string*. A partir disso, essa etapa utiliza a biblioteca Pandas com o objetivo de simplificar os dados para a criação das futuras métricas e análises.

A maior das transformações está direcionada exatamente na disposição das colunas. No caso da tabela de *commits*, as informações foram agrupadas em uma única coluna, sendo necessária a utilização da função *pandas.Series.str.strip* para separar os dados e tornar a tabela mais visível. O resultado dessas transformações seguindo o exemplo da biblioteca Pandas é o seguinte:

Figura 3 – Exemplo da tabela de *commits* após as transformações.

	author_name	id	...	insertions	message
0	Wes McKinney	9d0080576446de475d34b0dbb58389b15cd4f529	...	0	Initial directory structure.\n\ngit-svn-id: ht...
1	Wes McKinney	ec1a0a2a2571dc2c1c26612b374d4a66b22f0938	...	0	adding trunk\n\ngit-svn-id: http://pandas.goog...
2	Wes McKinney	1eeadf4e401647faa20911f531bc05c1872262ea	...	0	oops\n\ngit-svn-id: http://pandas.googlecode.c...
3	Wes McKinney	445114e1b20da8d4976c8d9050aa90c5bd508c54	...	0	added svn:ignore\n\ngit-svn-id: http://pandas....
4	Wes McKinney	c6b236db73ff81007909be6406f0e484edc4a9eb	...	21658	first commit with cleaned up code\n\ngit-svn-i...
...	...	...	...	...	...
33446	Mateusz Sokół	e61a0a8a9de56a2876a4468aed25b8a49608b660	...	14	MAINT: Partially revert `np.int_` changes (#55...
33447	Matheus Felipe	5bfb41f66388bfb46ad7025c954354562193c7f0	...	8	TYP/DOC: add HTMLFlavors type to read_html and...
33448	Patrick Hoefler	ea14a466799fe6c2ba42dc468cda5212ea431026	...	17	TST: Fix assert_is_sorted for eas (#55536)
33449	Paras Gupta	32c9c8feaf46d85119d66ddb09a4b69b28721c50	...	13	BUG: DataFrame.to_json OverflowError with np.I...
33450	jbrockmendel	746e5eee860b6e143c33c9b985e095dac2e42677	...	91	ENH: EA._from_scalars (#53089)\n\n* ENH: BaseS...

33451 rows × 7 columns

Fonte: Elaborada pelo autor.

Para a tabela de *trends*, a transformação ocorreu pois cada item da pesquisa (repositório no caso deste trabalho) estava distribuído como um item do cabeçalho, ficando inviável visualizar e analisar uma tabela com mais de oitenta e cinco colunas. A partir disso, foi utilizada a função *pandas.melt* para pivotar a tabela e deixá-la visualmente mais simples com a criação de uma coluna nomeada como *project*. O resultado dessas transformações seguindo o exemplo da biblioteca Pandas é o seguinte:

Figura 4 – Exemplo da tabela de *trends* após as transformações.

	date	project	relative_interest
0	2004-01-01	pandas	2
1	2004-02-01	pandas	1
2	2004-03-01	pandas	5
3	2004-04-01	pandas	0
4	2004-05-01	pandas	2
...	...	...	...
233	2023-06-01	pandas	67
234	2023-07-01	pandas	62
235	2023-08-01	pandas	67
236	2023-09-01	pandas	68
237	2023-10-01	pandas	68

238 rows × 3 columns

Fonte: Elaborada pelo autor.

4 Resultados e análises

Para analisar o ciclo de vida dos softwares, temos que levar em consideração que esses dados podem variar sem seguir um padrão específico, dependendo de fatores como a tendência de mercado, linguagem de programação, área de uso e atividade dos desenvolvedores. Neste trabalho, será realizada uma análise sobre a tendência de ascensão ou queda e correlação de cada *framework* e biblioteca, se baseando em duas métricas principais: MME e a correlação entre a quantidade de commits, de autores ativos e a pontuação do interesse relativo.

Para as análises de tendências, foram utilizadas MMEs de curtos e longos prazos. Isso faz com que um resultado de curta duração seja mais sensível a alterações nos dados do que o resultado de longa duração, como descrito na Seção 2.4.4. Para saber se a tendência é de alta ou baixa, basta comparar os dois valores de MMEs como exemplificado na Tabela 3 a seguir:

Tabela 3 – Comparação entre MMEs para indicação de tendência

Comparação	**Tendência**
MME de curto prazo $>$ MME de longo prazo	Alta
MME de curto prazo $<$ MME de longo prazo	Baixa

No presente trabalho foram utilizados períodos específicos para o cálculo da MME, sendo doze meses para o curto prazo e vinte e quatro meses para o longo prazo. Esses valores foram escolhidos com base no ciclo de vida de softwares e nas análises, pois com um valor menor para o curto prazo, os resultados poderiam ser alterados facilmente por aumentos e quedas repentinas, que podem não representar o comportamento geral ao longo do tempo.

A análise será dividida em duas etapas principais, uma quantitativa que mostra um panorama geral de como os resultados estão, e uma qualitativa, que exemplifica e demonstra alguns exemplos escolhidos com uma gama maior de detalhes. Além disso, será disponibilizado um repositório publico onde contém o código-fonte utilizado neste trabalho, possibilitando o usuário criar suas próprias análises [1].

4.1 Análises quantitativas

Essa primeira análise leva em conta todos os resultados finais para os repositórios selecionados, apresentando suas tendências e correlações respectivamente. Neste cenário não é levado em consideração os motivos e explicativas sobre cada um dos itens, por conta da grande quantidade de softwares escolhidos.

[1] <https://github.com/RonaldoRubens/Mineracao-de-Repositorios>

As tendências são aplicadas em três cenários, sobre a quantidade de *commits*, quantidade de autores ativos e o valor do interesse relativo. Todos os dados utilizados para esse cálculo tiveram como referência a ultima data em que existem dados para cada projeto, no caso de softwares ainda ativos, a última data foi 01/08/2023. Portanto, o cálculo é realizado se baseando no momento mais atual de cada item. A seguir é apresentada a Tabela 4 com todos os resultados de tendências obtidos:

Tabela 4 – Tendências gerais por projeto

Projeto	***Commit***	**Autor**	**Interesse**	**Geral**
armadillo-code	baixa	baixa	baixa	baixa
backbone	alta	baixa	baixa	baixa
bc-java	baixa	alta	baixa	baixa
botan	alta	alta	alta	alta
brain.js	baixa	baixa	alta	baixa
brakeman	baixa	baixa	alta	baixa
camping	alta	baixa	alta	alta
Catch2	baixa	baixa	baixa	baixa
Chart.js	baixa	baixa	alta	baixa
CNTK	baixa	alta	baixa	baixa
cppcms	baixa	alta	baixa	baixa
cpprestsdk	baixa	baixa	alta	baixa
cppunit	baixa	alta	baixa	baixa
cxf	alta	baixa	baixa	baixa
cypress	baixa	baixa	alta	baixa
d3	alta	alta	baixa	alta
dask	baixa	baixa	alta	baixa
deeplearning4j	baixa	baixa	alta	baixa
django	baixa	baixa	alta	baixa
eve	baixa	baixa	baixa	baixa
express	baixa	baixa	baixa	baixa
faces	baixa	baixa	baixa	baixa
fastapi	alta	alta	alta	alta
flask	alta	baixa	alta	alta
googletest	alta	alta	baixa	alta
grape	baixa	baixa	alta	baixa
h2o-3	baixa	baixa	baixa	baixa
hanami	baixa	baixa	alta	baixa
helmet	baixa	baixa	baixa	baixa
ITK	baixa	baixa	alta	baixa

Tabela 4 – Tendências gerais por projeto

Projeto	***Commit***	**Autor**	**Interesse**	**Geral**
jest	baixa	baixa	alta	baixa
jsrsasign	baixa	alta	alta	alta
jstat	baixa	baixa	baixa	baixa
junit4	baixa	baixa	alta	baixa
junit5	alta	alta	baixa	alta
koa	baixa	baixa	baixa	baixa
libtomcrypt	baixa	baixa	alta	baixa
mahout	baixa	baixa	baixa	baixa
minitest	baixa	baixa	baixa	baixa
mlpack	baixa	baixa	alta	baixa
moa	baixa	baixa	alta	baixa
nmatrix	baixa	baixa	baixa	baixa
node-restify	baixa	baixa	baixa	baixa
nsp	baixa	baixa	alta	baixa
openssl	alta	alta	alta	alta
orange3	baixa	baixa	alta	baixa
owasp-java-encoder	baixa	alta	baixa	baixa
pandas	alta	baixa	alta	alta
paramiko	alta	alta	alta	alta
pistache	baixa	baixa	alta	baixa
playframework	alta	baixa	baixa	baixa
pycrypto	alta	alta	alta	alta
PyTables	baixa	baixa	baixa	baixa
pytest	baixa	alta	alta	alta
pytorch	alta	alta	alta	alta
qunit	baixa	baixa	baixa	baixa
rails	alta	baixa	alta	alta
react	baixa	baixa	alta	baixa
restbed	baixa	baixa	alta	baixa
Resteasy	alta	baixa	baixa	baixa
robotframework	baixa	baixa	alta	baixa
roda	baixa	baixa	alta	baixa
rspec	baixa	baixa	baixa	baixa
ruby-fann	baixa	baixa	baixa	baixa
sciruby	baixa	baixa	alta	baixa
scruffy	alta	alta	baixa	alta

Tabela 4 – Tendências gerais por projeto

Projeto	***Commit***	**Autor**	**Interesse**	**Geral**
sinatra	alta	baixa	baixa	baixa
sorcery	baixa	baixa	alta	baixa
spark	alta	alta	alta	alta
spring-framework	alta	alta	baixa	alta
statsample	baixa	alta	alta	alta
struts	alta	alta	baixa	alta
struts1	baixa	baixa	baixa	baixa
synaptic	baixa	baixa	baixa	baixa
tensorflow	baixa	baixa	baixa	baixa
tensorflow.rb	baixa	baixa	baixa	baixa
test-unit	alta	alta	alta	alta
testng	baixa	baixa	baixa	baixa
tfjs	baixa	baixa	alta	baixa
unittest2	alta	baixa	baixa	baixa
vue	baixa	baixa	alta	baixa
web2py	baixa	baixa	baixa	baixa
weka-3.8	baixa	baixa	baixa	baixa
wt	alta	baixa	alta	alta
zaproxy	alta	baixa	alta	alta

As tendências consideradas como gerais são calculadas com base nas tendências de *commits*, autores e interesses, todas com o mesmo peso. Caso em algum projeto existam duas ou mais tendências de alta, a tendência geral também segue de alta. Da mesma forma segue para a tendência de baixa.

Com essa tabela, podemos analisar que, dos 85 projetos escolhidos, 23 tem uma tendência geral de alta, sendo:

- 18 projetos atuais
- 5 projetos legados

Dos 62 softwares com tendência geral de baixa, 27 deles tem tendência de interesse positivo, ou seja, podem ser projetos estáveis que não tem a necessidade atual de grandes modificações em código-fonte, com:

- 23 projetos atuais
- 4 projetos legados

Esse é o caso do *framework* Vue por exemplo, que é um dos líderes em criação de interfaces em JavaScript, altamente estável e que apresenta tendência geral de baixa. Por conta disso é de extrema importância uma análise mais detalhada, para verificar os valores gráficos e possíveis pesquisas complementares, disponíveis na Seção 4.2.

Para a correlação em *notebook* Jupyter, foram utilizados os modelos de Pearson e Spearman, pois como não há garantia de uma relação linear entre os dados obtidos, é importante considerar diferentes maneiras de calcular a métrica. Com uma análise prévia, foi possível identificar que o modelo com maior grau de assertividade é o de Spearman, pois não assume que os dados tenham distribuição normal, lidando com uma abrangência maior. Foram analisadas correlações entre: quantidade de *commits* pelo interesse relativo, quantidade de autores pelo interesse relativo e quantidade de *commits* pela quantidade de autores. A seguir é apresentada a Tabela 5 com todos os resultados de correlações obtidos:

Tabela 5 – Correlações gerais de Spearman por projeto

Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
armadillo-code	—	—	—
backbone	-0,219	0,234	0,785
bc-java	—	—	0,887
botan	-0,042	0,395	0,583
brain.js	0,406	0,578	0,888
brakeman	-0,086	0,266	0,789
camping	0,721	-0,327	-0,375
Catch2	0,233	0,592	0,493
Chart.js	-0,127	0,142	0,67
CNTK	—	—	0,936
cppcms	0,459	-0,216	0,095
cpprestsdk	-0,84	-0,282	0,434
cppunit	0,97	0,755	0,857
cxf	0,653	-0,133	0,369
cypress	-0,516	0,859	-0,31
d3	0,467	0,666	0,811
dask	-0,704	0,778	-0,447
deeplearning4j	0,171	0,668	0,529
django	-0,418	0,65	0,192
eve	0,124	-0,287	0,76
express	0,923	0,169	0,318
faces	-0,747	-0,807	0,918
fastapi	-0,119	0,842	0,061

Tabela 5 – Correlações gerais de Spearman por projeto

Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
flask	-0,287	0,049	0,103
googletest	-0,381	-0,389	0,923
grape	0,023	0,175	0,831
h2o-3	0,532	0,21	0,653
hanami	-0,24	-0,096	0,948
helmet	0,379	0,387	0,35
ITK	0,816	0,177	0,468
jest	-0,631	-0,729	0,91
jsrsasign	-0,286	-0,001	0,578
jstat	0,495	-0,776	-0,386
junit4	0,443	-0,259	0,602
junit5	-0,839	-0,216	0,391
koa	-0,765	-0,336	0,672
libtomcrypt	0,101	-0,176	0,894
mahout	0,355	0,704	0,722
minitest	0,058	-0,347	0,07
mlpack	-0,12	-0,242	0,843
moa	0,695	0,63	0,86
nmatrix	-0,575	0,503	0,12
node-restify	—	—	0,33
nsp	-0,569	-0,403	0,707
openssl	-0,6	-0,542	0,835
orange3	-0,128	0,051	0,901
owasp-java-encoder	—	—	0,572
pandas	0,513	0,921	0,544
paramiko	0,61	0,603	0,854
pistache	0,333	0,626	0,779
playframework	0,724	0,875	0,679
pycrypto	0,174	0,803	0,212
PyTables	0,232	-0,307	0,31
pytest	0,567	0,885	0,789
pytorch	0,952	0,953	0,972
qunit	0,388	0,864	0,547
rails	0,081	-0,113	0,87
react	-0,74	-0,643	0,727
restbed	-0,082	0,152	0,428

Tabela 5 – Correlações gerais de Spearman por projeto

Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
Resteasy	0,106	0,178	0,119
robotframework	-0,765	0,4	0,16
roda	0,001	0,297	0,693
rspec	-0,76	-0,011	0,246
ruby-fann	—	—	0,659
sciruby	0,176	0,149	-0,241
scruffy	—	—	-0,187
sinatra	—	—	0,563
sorcery	0,452	0,117	0,533
spark	0,199	0,623	0,665
spring-framework	-0,473	-0,917	0,708
statsample	0,372	-0,753	-0,199
struts	0,336	0,136	0,629
struts1	0,855	0,947	0,799
synaptic	0,933	-0,562	-0,407
tensorflow	0,659	0,609	0,982
tensorflow.rb	—	—	0,873
test-unit	-0,794	0,233	0,12
testng	-0,244	0,566	0,407
tfjs	-0,622	-0,458	0,821
unittest2	—	—	-0,44
vue	-0,428	-0,113	0,784
web2py	0,979	0,482	0,532
weka-3.8	—	—	-0,309
wt	-0,13	-0,481	-0,051
zaproxy	0,093	0,552	0,373

Dentre os resultados, é possível quantificar projetos que possuem correlações altas e baixas, com valores maiores que 0,5 e menores que -0,5 para as correlações altas e valores entre -0,5 e 0,5 para correlações baixas, como exemplificado na Tabela 6 a seguir:

Tabela 6 – Quantidade de projetos por agrupamento

	Commit/Interesse	**Autor/Interesse**	**Commit/Autor**
$>$ 0,5	16	25	50
$<$ -0,5	15	8	0
Entre -0,5 e 0,5	43	41	35
Sem dados	11	11	0

Com isso, é possível identificar um padrão entre *commits* e autores, que não apresentam valores negativos e tem o maior percentual de correlações fortes entre as comparações. Isso é natural pois uma maior quantidade de autores normalmente implica em uma maior quantidade de *commits*. Já as relações que envolvem o interesse relativo, algumas tem valor negativo tanto para *commits* quanto para autores, o que pode auxiliar na identificação de projetos já estáveis, que tem tendência de baixa para modificações em código e alta para interesse da comunidade, ou também pode representar uma parcela de projetos legados, que já não são mais atualizados porém ainda são utilizados em algumas aplicações.

O projeto NSP (Node Security Protect) é um bom exemplo, pois tem alta tendência de interesse relativo, baixa tendência em *commits* e autores e correlações referentes ao interesse negativas. Neste caso o NSP é um software muito utilizado no mercado, com seu código-fonte estável, não tem a necessidade atual de grandes modificações.

Na Tabela 5 existem resultados em branco por falta de informações suficientes dos respectivos softwares.

4.2 Análise qualitativa

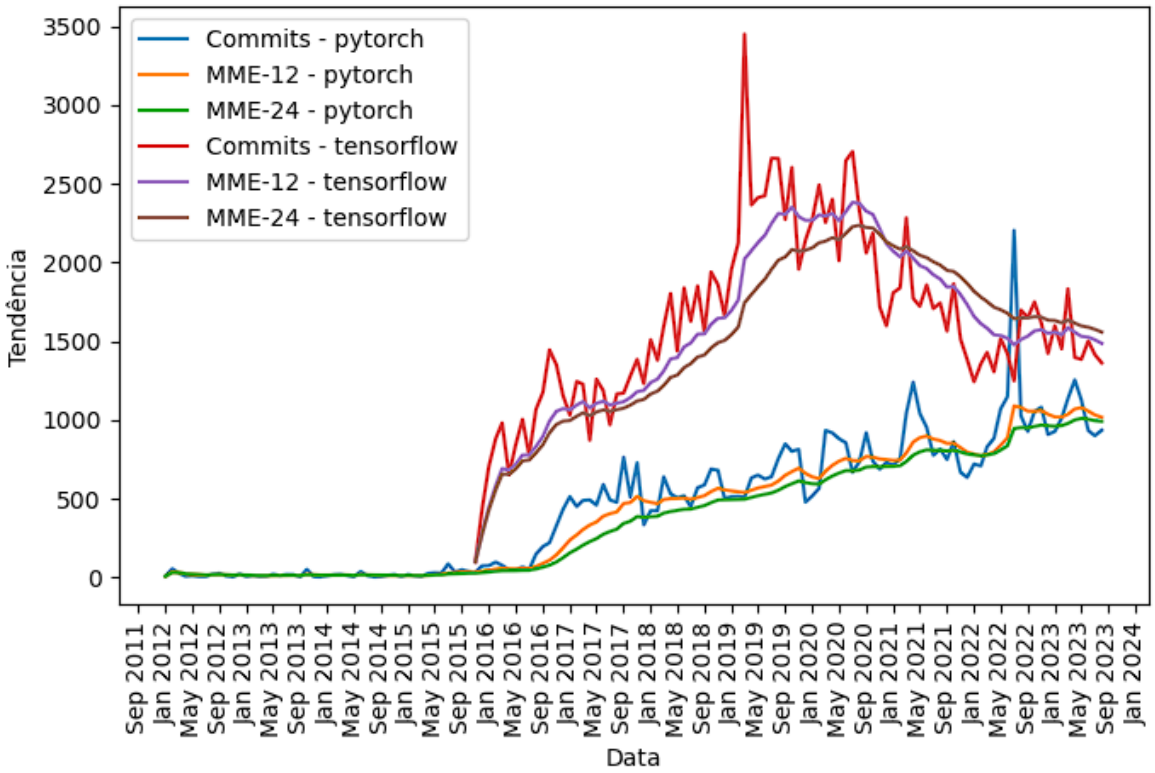
A análise qualitativa é mais exploratória quando comparada à análise quantitativa, se aproximando de uma possível situação real no início do desenvolvimento de uma nova aplicação, onde os resultados deste trabalho e pesquisas complementares serão utilizadas para identificar o software que mais trará valor para o projeto. Como a quantidade total de resultados de tendências e correlações é muito abrangente, foram escolhidas quatro análises pontuais que tem o objetivo de demonstrar mais a fundo cada métrica alcançada neste trabalho, com suas devidas justificativas e explicações.

4.2.1 PyTorch x TensorFlow

O PyTorch e o TensorFlow são um exemplo de projetos "concorrentes", que abordam a área de aprendizado de máquina, ambos utilizados na linguagem Python. Segundo os dados de *commit*, o PyTorch teve o começo das suas atividades em código fonte no início de 2012, porém o seu lançamento foi realizado somente em setembro de 2016. Por conta disso ele é considerado mais novo em comparação com o TensorFlow, com lançamento em novembro de 2015.

Para análise inicial, é importante visualizar as tendências sobre o número de *commits*, autores ativos e de interesse relativo para entender como esses dados se relacionam e obter informações que podem ser importantes para uma eventual decisão de projeto. A apresentação dessas tendências estão nas figuras 5, 6 e 7 respectivamente.

Figura 5 – Tendência de *commits* entre PyTorch e TensorFlow.



Fonte: Elaborada pelo autor.

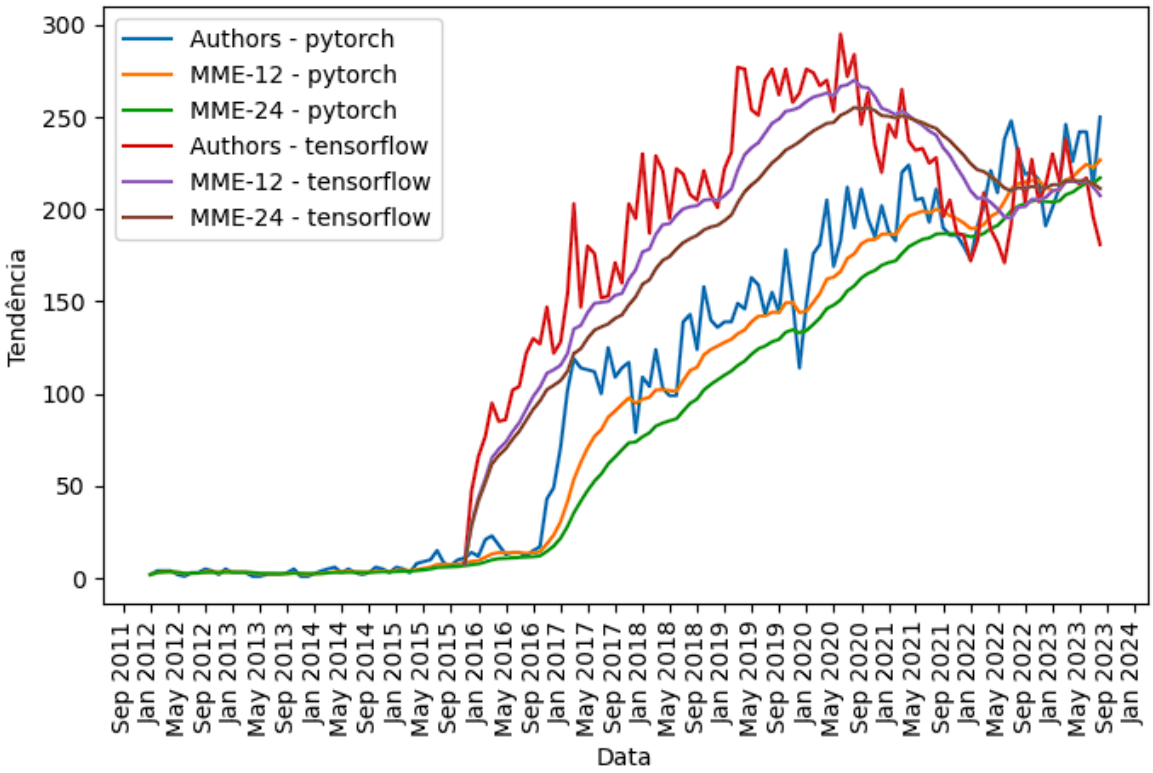
A tendência do PyTorch em quantidade de *commits* é de alta, com as seguintes métricas:

- MME de curto prazo = 1017,15
- MME de longo prazo = 1004,10

Já a tendência do TensorFlow em quantidade de *commits* é de baixa, com as seguintes métricas:

- MME de curto prazo = 1486,54
- MME de longo prazo = 1533,46

Figura 6 – Tendência de autores entre PyTorch e TensorFlow.



Fonte: Elaborada pelo autor.

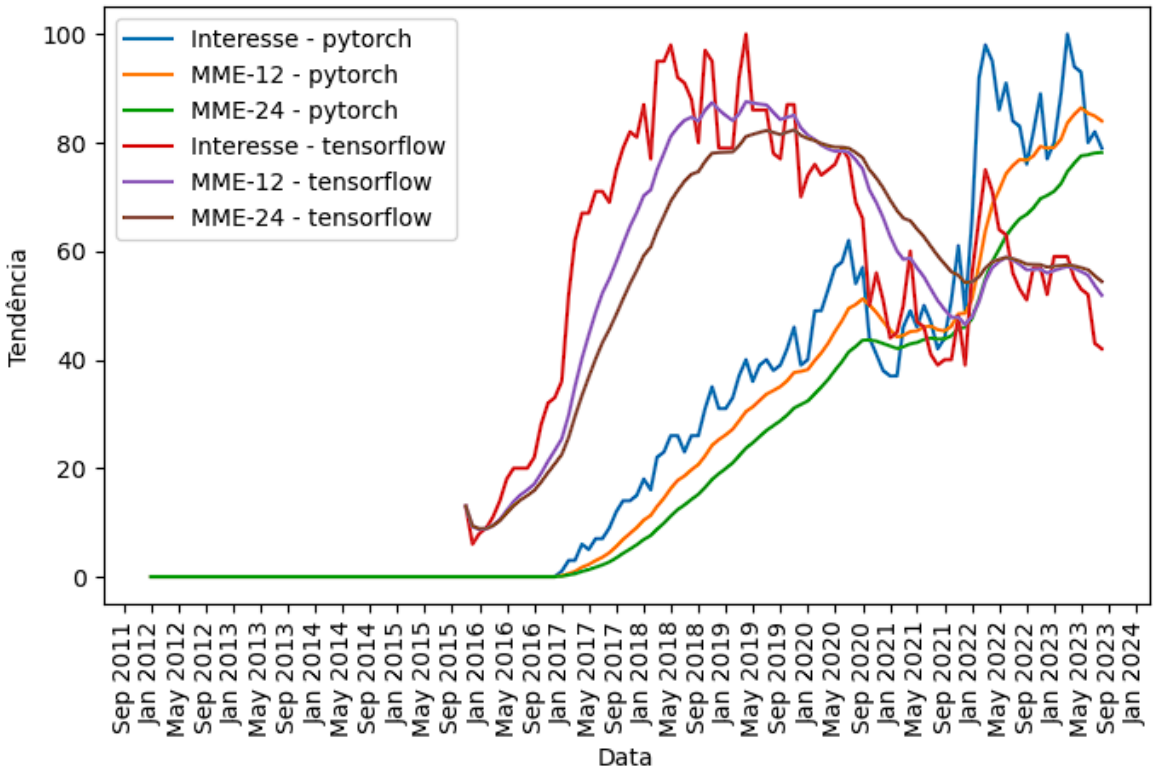
A tendência do PyTorch em quantidade de autores é de alta, com os resultados:

- MME de curto prazo = 226,74
- MME de longo prazo = 220,21

Já a tendência do TensorFlow em quantidade de autores é de baixa, com os resultados:

- MME de curto prazo = 207,49
- MME de longo prazo = 209,97

Figura 7 – Tendência de interesse entre PyTorch e TensorFlow.



Fonte: Elaborada pelo autor.

A tendência do PyTorch em relação a quantidade de interesse é de alta, com os seguintes resultados:

- MME de curto prazo = 83,99
- MME de longo prazo = 80,37

Já a tendência do TensorFlow em relação a quantidade de interesse é de baixa, com os seguintes resultados:

- MME de curto prazo = 51,85
- MME de longo prazo = 53,65

Com isso, é possível observar uma vantagem das tendências do PyTorch em relação ao TensorFlow. De acordo com o (DAI et al., 2022), o PyTorch é um software mais fácil de utilizar e tem uma curva de aprendizado mais rápida, o que ajuda a explicar uma certa vantagem no interesse recente. Apesar de ambos serem projetos bem estabelecidos no mercado, o TensorFlow teve um crescimento no número de *commits* e autores mais acentuado nos anos iniciais quando comparado ao seu concorrente, se tornando normal uma diminuição desse fluxo após aproximadamente 4 anos.

A correlação entre as métricas analisadas é devidamente proporcional e esperada, com os seguintes resultados mostrados na Tabela 7:

Tabela 7 – Correlações dos projetos PyTorch e TensorFlow.

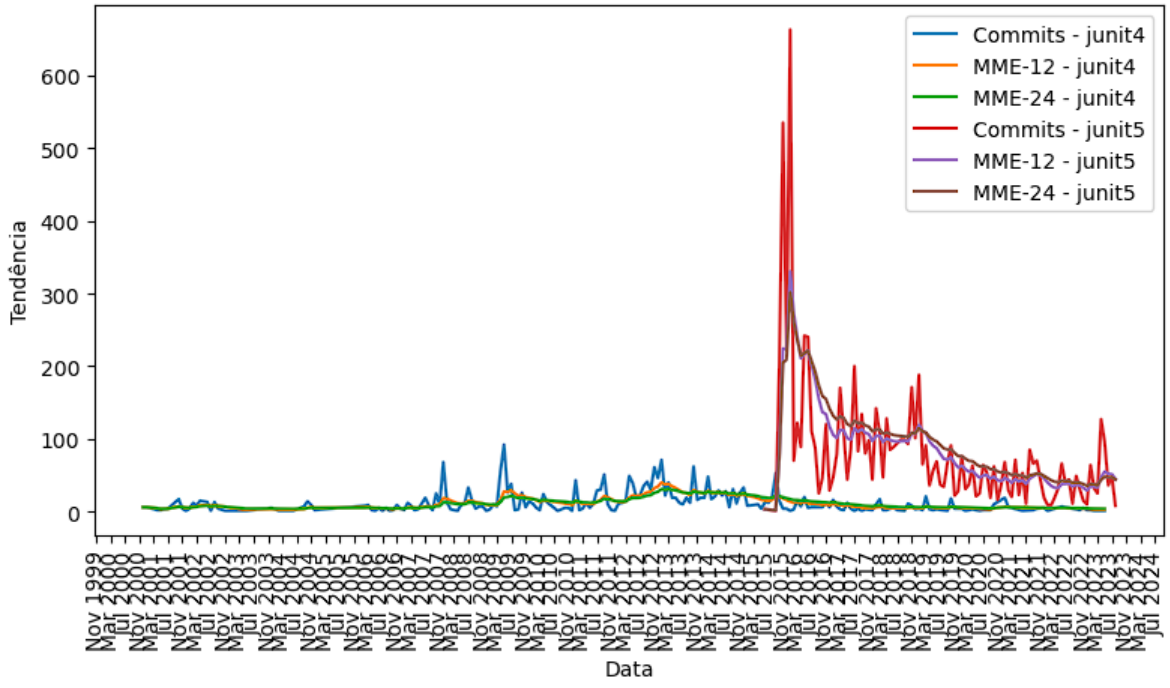
Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
pytorch	0,952	0,953	0,972
tensorflow	0,659	0,609	0,982

4.2.2 Junit4 x Junit5

A biblioteca Junit5 é uma versão seguinte do software Junit4, portanto ambos são ferramentas para automatização de testes unitários, utilizados na linguagem Java. Ao observar o código fonte e o início dos *commits*, é natural ter uma grande diferença entre as datas, sendo janeiro de 2001 para a versão anterior do projeto e maio de 2015 para a versão atual.

São obtidas as seguintes tendências sobre *commits*, autores ativos e interesse relativo para esses *frameworks* nas figuras 8, 9 e 10 respectivamente:

Figura 8 – Tendência de *commits* entre Junit4 e Junit5.



Fonte: Elaborada pelo autor.

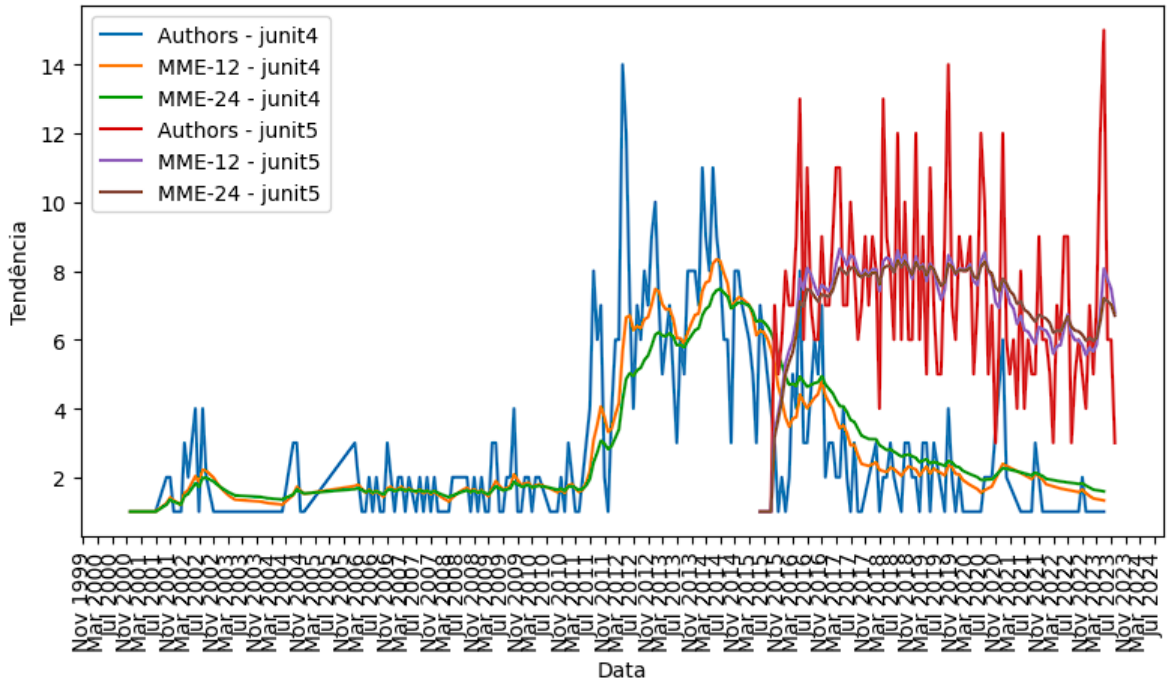
A tendência do Junit5 em quantidade de *commits* é de alta, com as seguintes métricas:

- MME de curto prazo = 44,74
- MME de longo prazo = 43,90

Já a tendência do Junit4 em quantidade de *commits* é de baixa, com as seguintes métricas:

- MME de curto prazo = 2,95
- MME de longo prazo = 3,84

Figura 9 – Tendência de autores entre Junit4 e Junit5.



Fonte: Elaborada pelo autor.

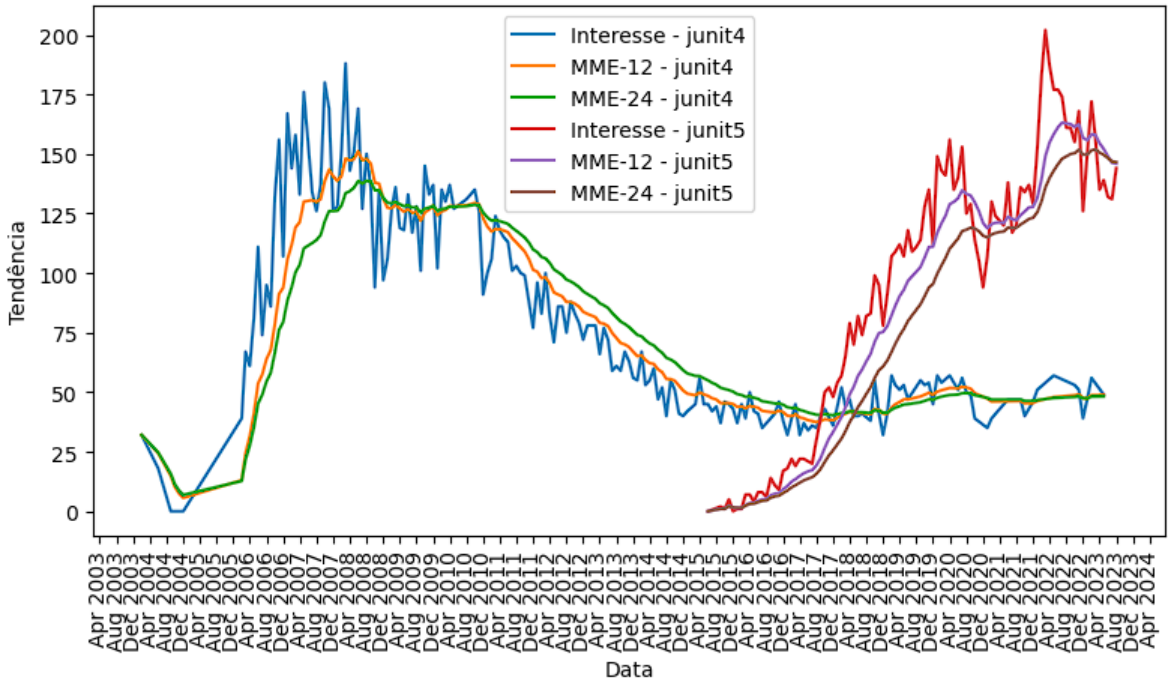
A tendência do Junit5 em quantidade de autores é de alta, com os resultados:

- MME de curto prazo = 6,79
- MME de longo prazo = 6,69

Já a tendência do Junit4 em quantidade de autores é de baixa, com os resultados:

- MME de curto prazo = 1,32
- MME de longo prazo = 1,59

Figura 10 – Tendência de interesse entre Junit4 e Junit5.



Fonte: Elaborada pelo autor.

A tendência do Junit5 em relação a quantidade de interesse é de baixa, com os seguintes resultados:

- MME de curto prazo = 145,90
- MME de longo prazo = 146,64

Já a tendência do Junit4 em relação a quantidade de interesse é de alta, com os seguintes resultados:

- MME de curto prazo = 48,79
- MME de longo prazo = 48,20

Como o Junit5 é uma versão mais atual do Junit4, é natural que as tendências obtidas para *commits* e autores sejam de altas para o projeto atual e baixas para o antecessor, caso o Junit5 ainda seja ainda o mais recente. Porém, o resultado referente a tendência de interesse relativo foi de alta na versão anterior, contrariando a ordem natural do ciclo de vida dos softwares.

Isso pode ser explicado pelo fato de que o Junit4, apesar do longo tempo desde o lançamento, é utilizado em muitos projetos que ainda não foram atualizados para a nova versão, seja pela falta de manutenção, ou pela incompatibilidade do Junit5 com alguns softwares

utilizados anteriormente (GARCIA, 2017). Analisando os valores de interesse, podemos ver porém que o Junit5 é quase 3x mais procurado quando comparado com a versão anterior, apesar de sua tendência estar em baixa.

Essa inversão de expectativa causada pelo interesse relativo é também refletida no cálculo das correlações, que são inversamente proporcionais para o Junit4 entre a quantidade de autores e interesse e também para o Junit5 entre a quantidade de *commits*, autores e interesse mostradas na Tabela 8:

Tabela 8 – Correlações dos projetos Junit4 e Junit5.

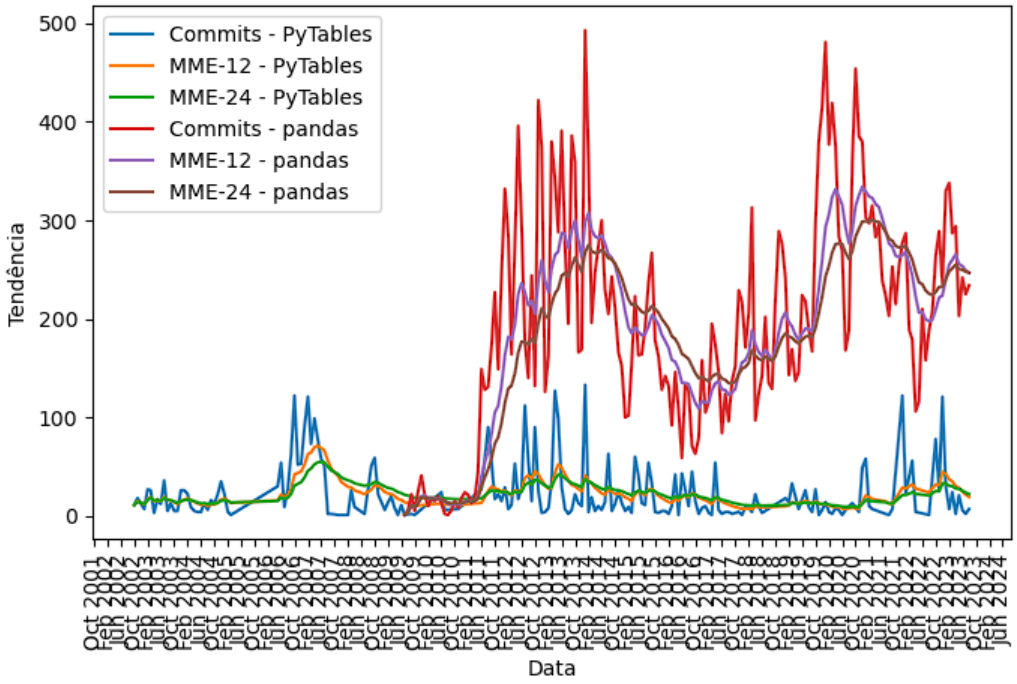
Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
Junit4	0,443	-0,259	0,602
Junit5	-0,839	0,216	0,391

4.2.3 Pandas x PyTables

Avaliar as bibliotecas Pandas e PyTables possibilita a obtenção de *insights* importantes, pois o PyTables é um projeto legado, que já tem o seu ciclo de vida muito bem definido e quase encerrado. Já o Pandas é um projeto amplamente usado e ativo pela comunidade, com o seu ciclo de vida ainda em andamento. Ambos abordam a área de ciência de dados, utilizados na linguagem Python.

Para esses softwares são apresentadas as seguintes tendências sobre *commits*, autores ativos e interesse relativo nas figuras 11, 12 e 13 respectivamente:

Figura 11 – Tendência de *commits* entre Pandas e PyTables.



Fonte: Elaborada pelo autor.

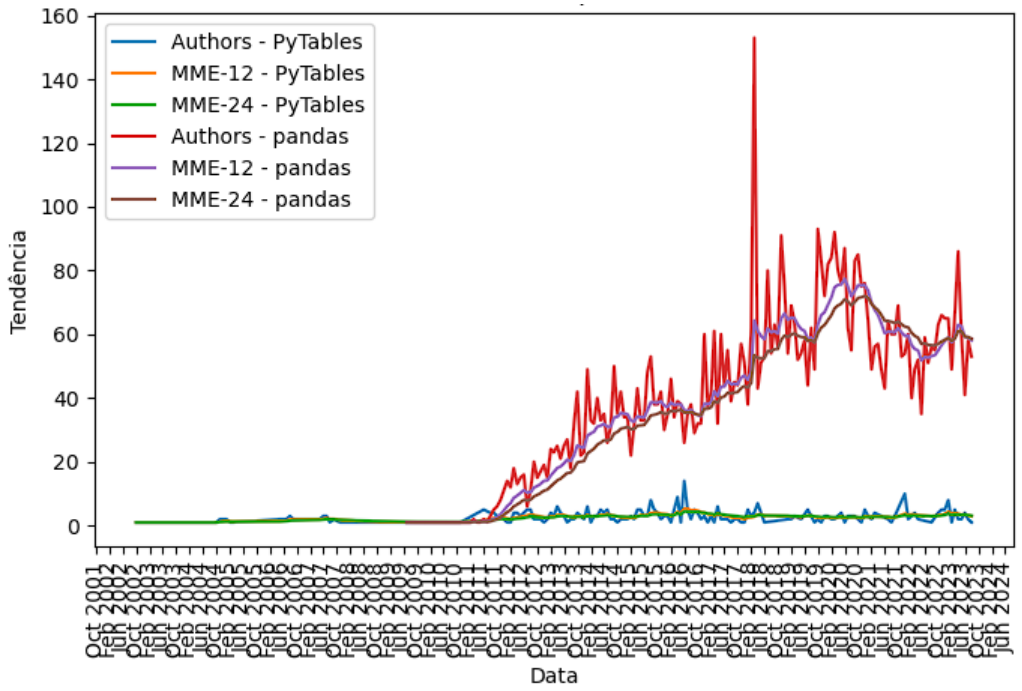
A tendência do Pandas em quantidade de *commits* é de alta, com as seguintes métricas:

- MME de curto prazo = 246,85
- MME de longo prazo = 246,79

Já a tendência do PyTables em quantidade de *commits* é de baixa, com as seguintes métricas:

- MME de curto prazo = 19,31
- MME de longo prazo = 22,34

Figura 12 – Tendência de autores entre Pandas e PyTables.



Fonte: Elaborada pelo autor.

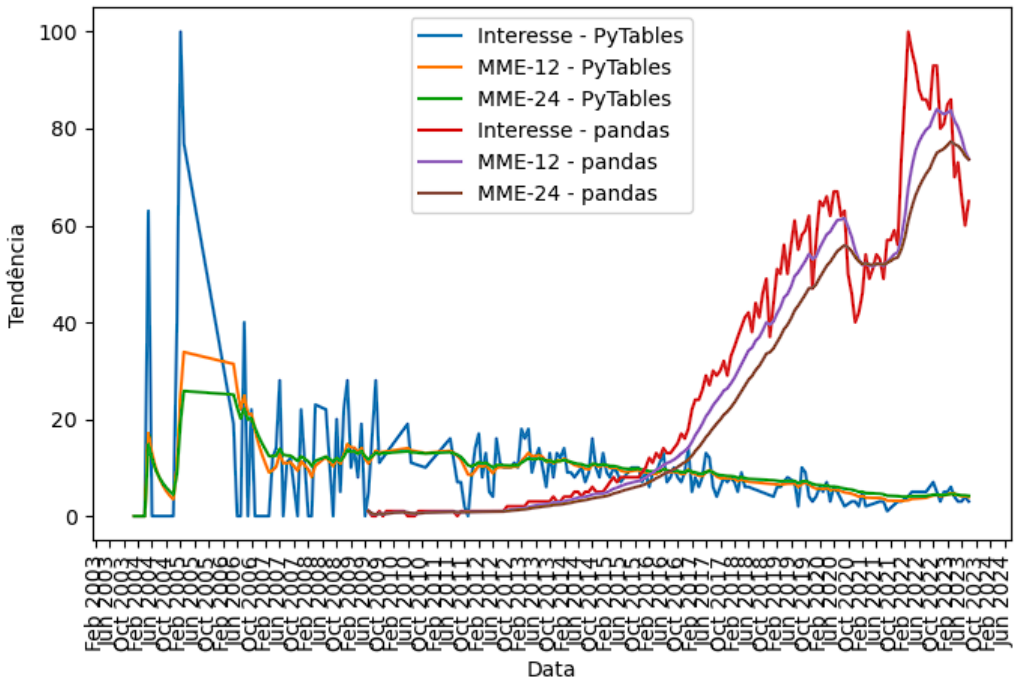
A tendência do Pandas em quantidade de autores é de alta, com os resultados:

- MME de curto prazo = 58,08
- MME de longo prazo = 58,63

Já a tendência do PyTables em quantidade de autores é de baixa, com os resultados:

- MME de curto prazo = 2,92
- MME de longo prazo = 3,11

Figura 13 – Tendência de interesse entre Pandas e PyTables.



Fonte: Elaborada pelo autor.

A tendência do Pandas em relação a quantidade de interesse é de alta, com os seguintes resultados:

- MME de curto prazo = 73,72
- MME de longo prazo = 73,61

Já a tendência do PyTables em relação a quantidade de interesse é de baixa, com os seguintes resultados:

- MME de curto prazo = 3,92
- MME de longo prazo = 4,15

As tendências da biblioteca Pandas estão em alta, porém por uma margem muito pequena, o que pode indicar que o projeto está em um dos seus pontos de maior estabilidade, tendo já atingido níveis elevados em numero de *commits*, autores e interesse. Ao passar do tempo, o natural para o projeto é diminuir o numero de commits e autores, se não houver uma grande mudança estrutural no código, e manter o interesse relativo até que acabe a sua manutenção ou surja uma biblioteca que torne o Pandas obsoleto.

O ciclo de vida completo do PyTables é evidente, apesar de ser um projeto não totalmente descontinuado, apresentando picos de interesse em seus anos iniciais, antecipando

uma grande queda até o atual momento. Como é um software que ainda tem uma pequena atividade, principalmente para manuseio de grandes conjuntos de dados (PYTABLES, 2022), o PyTables tem constância no numero de *commits* e autores, porém sem nenhuma expectativa de crescimento.

O único ponto diferente nas correlações se dá pela relação inversamente proporcional entre autores e interesse relativo por parte do PyTables na Tabela 9:

Tabela 9 – Correlações dos projetos Pandas e PyTables.

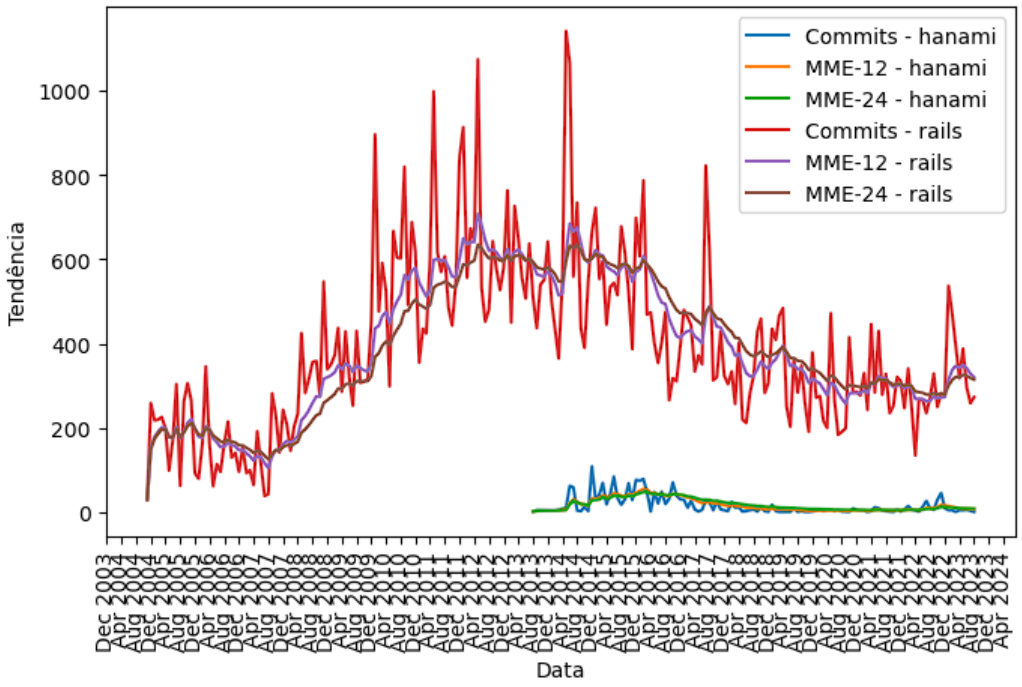
Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
pandas	0,513	0,921	0,544
PyTables	0,232	0,307	0,31

4.2.4 Ruby on Rails (RoR) x Hanami

Os projetos RoR e Hanami são um outro bom exemplo de projetos "concorrentes", abordando a área de desenvolvimento web na linguagem Ruby. Apesar de serem amplamente comparados, o Hanami é mais recente e moderno, tendo o início das suas atividades em 2014, dez anos de diferença frente ao RoR.

Para esses projetos são obtidas as seguintes tendências sobre *commits*, autores ativos e interesse relativo nas figuras 14, 15 e 16 respectivamente:

Figura 14 – Tendência de *commits* entre Ruby on Rails e Hanami.



Fonte: Elaborada pelo autor.

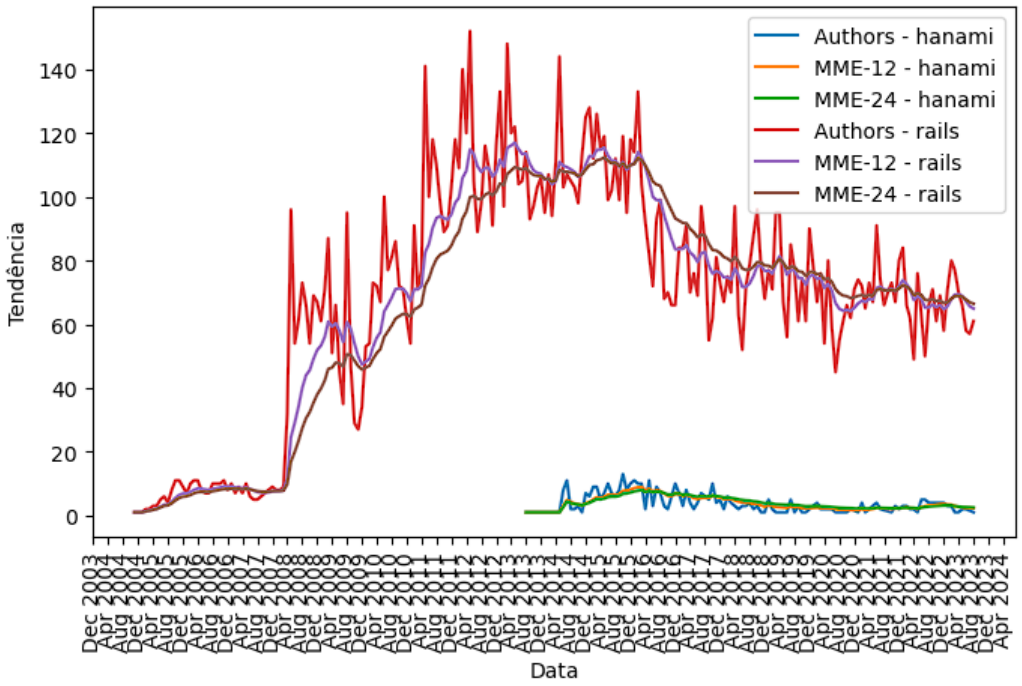
A tendência do RoR em quantidade de *commits* é de alta, com as seguintes métricas:

- MME de curto prazo = 320,09
- MME de longo prazo = 314,84

Já a tendência do Hanami em quantidade de *commits* é de baixa, com as seguintes métricas:

- MME de curto prazo = 7,75
- MME de longo prazo = 8,68

Figura 15 – Tendência de autores entre Ruby on Rails e Hanami.



Fonte: Elaborada pelo autor.

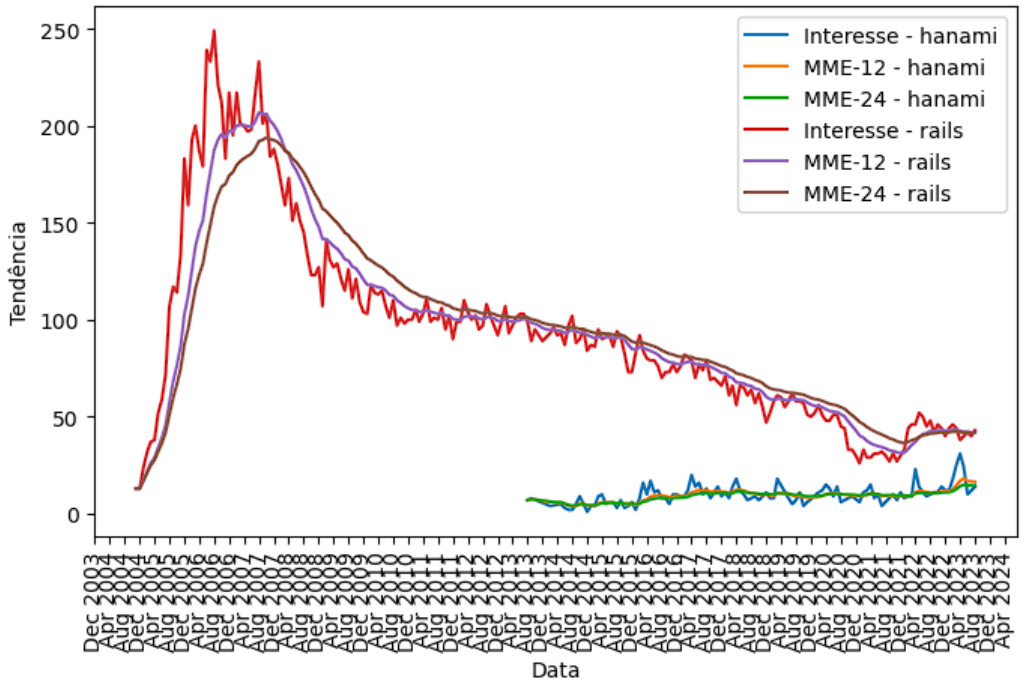
A tendência do RoR em quantidade de autores é de baixa, com os resultados:

- MME de curto prazo = 64,92
- MME de longo prazo = 66,44

A tendência do Hanami em quantidade de autores também é de baixa, com os resultados:

- MME de curto prazo = 2,27
- MME de longo prazo = 2,52

Figura 16 – Tendência de interesse entre Ruby on Rails e Hanami.



Fonte: Elaborada pelo autor.

A tendência do RoR em relação a quantidade de interesse é de alta, com os seguintes resultados:

- MME de curto prazo = 42,02
- MME de longo prazo = 41,85

A tendência do Hanami em relação a quantidade de interesse também é de alta, com os seguintes resultados:

- MME de curto prazo = 16,41
- MME de longo prazo = 14,49

Apesar do longo tempo desde o lançamento, o Ruby on Rails segue sendo líder em aplicações web em Ruby, por conta da sua estabilidade e eficiência na maioria das aplicações envolvidas. O Hanami é um projeto ainda em ascensão e recente, que conta com menos apoio da comunidade, implementações e tutoriais disponíveis quando comparado ao RoR (BUSZMAN, 2019). Isso fica evidente na análise das tendências e das métricas, com maior quantidade de *commits*, autores e interesse por parte do software mais antigo. Porém, o Hanami tem tendência de alta para o interesse relativo, pois é promissor, flexível e conta com ótimas estruturas de códigos e documentações.

As correlações entre as métricas tem o seguinte resultado na Tabela 10:

Tabela 10 – Correlações dos projetos Ruby on Rails e Hanami.

Projeto	**Commit/Interesse**	**Autor/Interesse**	**Commit/Autor**
rails	0,081	0,113	0,87
hanami	0,24	0,096	0,948

4.3 Considerações

As análises realizadas na Seção 4.2 apresentam um cenário real que auxilia na escolha de um novo software em um projeto. Além dos resultados de tendências e correlações sobre a quantidade de *commits*, autores e interesse relativo, é possível observar pontos importantes nos gráficos como cada quantidade e distribuição dos dados ao longo do tempo. Pesquisas auxiliares são utilizadas para fundamentar e complementar a análise a fim de obter resultados mais coesos.

O PyTorch e o TensorFlow são bons exemplos de que os dados apresentados podem realmente auxiliar nessa seleção de uma nova ferramenta, como mostrado na Seção 4.2.1. É perceptível pelas tendências que o PyTorch leva certa vantagem em relação ao TensorFlow, sendo um projeto com maior possibilidade de crescimento e manutenção. Com a ajuda das pesquisas complementares, foi possível observar também que o PyTorch é um software mais fácil de ser aprendido e utilizado, fundamentando ainda mais a sua escolha.

Portanto, cada análise pode ter nuances únicas e características especificas, dependendo de diversos fatores como idade de cada software, engajamento da comunidade e variações de mercado, porém é inevitável o benefício de ter ações baseadas em dados concretos, que encaminham e norteiam a tomada de decisão.

5 Conclusão

Este trabalho oferece uma abordagem complementar para a análise do ciclo de vida de softwares, com foco em desenvolvedores que buscam fundamentar suas escolhas de *frameworks* e bibliotecas no início de um novo projeto. Ao examinarmos os resultados de tendências e correlações para o número de *commits*, autores e de interesse relativo, ganhamos *insights* valiosos sobre a continuidade, manutenibilidade e o engajamento da comunidade em torno de um determinado software.

Essas informações são importantes para entender não apenas o presente, mas também antecipar o caminho futuro do projeto e capitalizar boas oportunidades proativamente. Dessa forma, a escolha dos elementos tecnológicos se torna estratégica, resultando em projetos mais robustos, adaptáveis e alinhados com as necessidades do mercado.

Dentro desse projeto, há também a possibilidade futura de adição de novas fontes de dados e métricas de análise, por exemplo a obtenção de dados do StackOverflow para entender melhor como a comunidade se comporta frente a um projeto, implementação de algoritmos de *machine learning* e a utilização de mais informações de *commits* nas análises, como quantidade de linhas adicionadas e deletadas.

Portanto, a aplicação da mineração de repositórios como uma ferramenta de análise de ciclo de vida de software representa um passo significativo em direção a um desenvolvimento mais informado e eficaz, onde as tendências convergem para guiar o desenvolvimento por meio dos dados.

Referências

BUSZMAN, M. Comparação de frameworks ruby modernos: Ror vs hanami. *Netguru*, 2019. Acesso em: 20 out. 2023. Disponível em: <https://www.netguru.com/blog/comparison-ror-vs-hanami>.

DAI, H.; PENG, X.; SHI, X.; HE, L.; XIONG, Q.; JIN, H. Reveal training performance mystery between tensorflow and pytorch in the single gpu environment. *Science China Information Sciences*, Springer, v. 65, p. 1–17, 2022.

DUARTE, N. F. B. *Frameworks e Bibliotecas Javascript*. Tese (Doutorado) — Instituto Politecnico do Porto (Portugal), 2015.

ELDER, A. Aprenda a operar no mercado de ações. *Rio de Janeiro: Editora Campus*, 2006.

GARCIA, B. *Mastering Software Testing with JUnit 5: Comprehensive guide to develop high quality Java applications*. [S.l.]: Packt Publishing Ltd, 2017.

HASSAN, A. E. The road ahead for mining software repositories. In: IEEE. *2008 frontiers of software maintenance*. [S.l.], 2008. p. 48–57.

JUPYTER. 2023. Acesso em: 08 out. 2023. Disponível em: <https://docs.jupyter.org/en/latest/>.

MALDONADO, J. C.; BRAGA, R. T. V.; GERMANO, F. S. R.; MASIERO, P. C. Padrões e frameworks de software. *Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, SP, Brasil*, p. 28, 2002.

MATPLOTLIB. 2023. Acesso em: 08 out. 2023. Disponível em: <https://matplotlib.org/stable/users/index.html>.

PANDAS. 2023. Acesso em: 08 out. 2023. Disponível em: <https://pandas.pydata.org/docs/index.html>.

PYDRILLER. 2023. Acesso em: 08 out. 2023. Disponível em: <https://pydriller.readthedocs.io/en/latest/intro.html#>.

PYTABLES. 2022. Acesso em: 20 out. 2023. Disponível em: <https://www.pytables.org/>.

PYTHON. 2023. Acesso em: 08 out. 2023. Disponível em: <https://docs.python.org/3/>.

PYTRENDS. 2023. Acesso em: 08 out. 2023. Disponível em: <https://pypi.org/project/pytrends/>.

SOUSA, Á. Coeficiente de correlação de pearson e coeficiente de correlação de spearman: o que medem e em que situações devem ser utilizados? *Correio dos Açores*, Gráfica Açoreana, Lda, p. 19–19, 2019.