# Abstract

## Background

The healthcare industry is seeing a digital revolution, resulting in an ever-growing influx of data. This transformation creates an urgent need for efficient and automated data visualization tools. Visual Viper (VV) aims to meet this demand by offering an automated Python library that streamlines the complex and often time-consuming process of creating data visualizations for scientific communication.

## Aim

The aim of this study is to outline the development of VV, assess its performance and adaptability, explore its modular design and development methodologies, and establish its practical applications in healthcare research.

## Methods

Built using Python, VV employs Vega-Lite for high-level interactive graphics. The library is structured with modular, extensible architecture, developed with object-oriented programming (OOP) and test-driven development (TDD) practices. Docker containerization ensures a consistent development environment, and GitLab version control, aligned with Semantic Versioning, streamlines collaborative development. Native CI/CD capabilities of GitLab further enrich the development process. VV operates environment-agnostically and offers serverless deployment options.

## Results

VV includes various interconnected components, each responsible for specific tasks ranging from data retrieval to chart rendering. Four main classes ('DatasetBuilder', 'ChartNotationBuilder', ‘ChartRenderer’, and 'ChartDeployer') encapsulate the respective functionalities, thus aiding in code maintenance and extension.

Evaluation metrics, captured using Monday.com and Python’s time library, showed that while VV required a longer initial setup time (2h vs. 0.5h), it outperformed manual methods in "Time-to-Final-Chart" (2h9min vs. 14h54min) for a project involving 72 spreadsheets. Adjusted metrics accounting for task fatigue and human intervention also favor VV, especially for larger and ongoing projects.

VV effectively minimizes manual labor, ensures data visualization consistency and fosters best practices in scientific communication. Current limitations include a focus on mostly specific organizational workflows and visualizations.

## Conclusion

VV presents a robust, automated, and customizable solution for data visualization. It holds promise for significantly enhancing scientific communication efficiency within the healthcare sector, with its modular and scalable design paving the way for future developments.

# Resumo

## Contexto

O sector da saúde está a ser alvo de uma revolução digital que resulta no aumento crescente de dados. Esta transformação cria uma necessidade urgente de ferramentas de visualização de dados eficientes e automatizadas. Visual Viper (VV) tem em vista responder a esta necessidade, oferecendo uma biblioteca Python que automatiza e simplifica o processo complexo e muitas vezes demorado de criação de visualizações de dados para comunicação científica.

## Objetivo

O objetivo deste trabalho é descrever o desenvolvimento do VV, avaliar o seu desempenho e adaptabilidade, explorar o seu desenho modular e metodologias de desenvolvimento utilizadas, bem como estabelecer as suas aplicações práticas na investigação em saúde.

## Métodos

Construído com recurso à linguagem Python, o VV emprega Vega-Lite para gráficos interativos de alto nível. A biblioteca é estruturada com arquitetura modular, extensível, desenvolvida com práticas de programação orientada a objetos (OOP) e desenvolvimento orientado por testes (TDD). A utilização de contentores Docker garante um ambiente de desenvolvimento consistente, e o controle de versão utilizando GitLab em conjugação com o sistema de Versionamento Semântico, simplifica o desenvolvimento colaborativo. As capacidades nativas de CI/CD do GitLab enriquecem ainda mais o processo de desenvolvimento. O VV opera de forma agnóstica de ambiente e permite opções de implementação sem servidor.

## Resultados

O VV inclui vários componentes interconectados, cada um responsável por tarefas específicas que vão desde a leitura de dados até à renderização de gráficos. Quatro classes principais - 'DatasetBuilder', 'ChartNotationBuilder', ‘ChartRenderer’, e 'ChartDeployer' - encapsulam as respectivas funcionalidades, facilitando a manutenção e extensão do código.

As métricas de avaliação, capturadas com recurso ao Monday.com e a biblioteca ‘time’ do Python, mostraram que embora o VV tenha exigido um tempo de configuração inicial mais longo (2h vs. 0.5h), superou os métodos manuais em "Time-to-Final-Chart" (2h9min vs. 14h54min) para um projeto que envolvia 72 folhas de cálculo. Métricas ajustadas que incluem o efeito da fadiga da tarefa e a intervenção humana, também favorecem o VV, especialmente para projetos maiores e contínuos.

O VV efetivamente minimiza o trabalho manual, garante a consistência da visualização dos dados, e promove boas práticas de comunicação científica. Atualmente, as limitações incluem um foco em fluxos de trabalho e visualizações tendencialmente específicas da organização.

## Conclusão

O VV apresenta uma solução robusta, automatizada e personalizável para a visualização de dados. Promete melhorar significativamente a eficiência da comunicação científica dentro do setor de saúde, com seu design modular e escalável abrindo caminho para desenvolvimentos futuros.