Towards Reproducibility and Knowledge Transfer in AI-assisted Data Analysis Code Generation

Robert Haase[a,b,*]

[a]Data Science Center, Leipzig University, Humboldtstraße 25, 04105 Leipzig, Germany
[b]Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden / Leipzig
ORCID (Robert Haase): https://orcid.org/0000-0001-5949-2327

Abstract.
Abstract will be added later

1 Introduction

Generative artificial intelligence and Large-language models (LLMs) in particular are changing the way we do data science. Most prominently, scientists use the technology for generating data analysis code [3, 4] and [re-]writing text. Prompts are commonly not stored so that knowledge transfer between scientists on how to prompt efficiently and responsibly is hindered. At the time of publication of scientific code and manuscripts, it may be hard to reproduce the scientific workflow that lead to a given set of data analysis code and to identify the parts of the project which were implemented by the human and the parts that were created by an LLM. A professional peer-review system documenting which human read and confirmed LLM-written code is not established in contemporary scientific culture. Such systems do exist for collaborative code editing involving multiple humans. E.g. the github.com online platform is well-established in the open-source community for discussing issues, for sharing potential solutions, and for reviewing code. It was also shown before that LLMs can solve real-world GitHub issues [1]. I am presenting git-bob, a practical implementation of an LLM-based assistant that can answer to GitHub issues, discuss potential solutions with humans iteratively, write code for them, and submit it as pull-request to reviewed by humans. With such a tool, interaction between humans and LLMs can be documented and later reproduced why data analysis is written in a certain way, if it was written by a human or an LLM, and if it was generated by an LLM, which human reviewed the code before it became part of the entire solution. In a world where scientists demonstrate that LLM-based systems can write entire papers, review and improve them [2] independently, such a solution is urgently need to be established as part of good scientific practice. git-bob is available open-source: https://github.com/haesleinhuepf/git-bob

2 Features and limitations

A common workflow, demonstrated in Figure 1A, is that a user opens an issue on a repository on Github.com, where git-bob is installed. If the user is repository member, they can trigger git-bob to answer. If they are externals, a repository member has to do this. git-bob may then answer the question, potentially including a code snippet. Multiple users and the AI-assistant can then argue back and forth until some potential solution is reached. Git-bob can then formulate a GitHub pull-request, e.g. with a Jupyter Notebook containing the entire solution to a given issue. A human needs to review this pull-request and merge it into the code base of the repository. Git-bob also has the capability to review pull-requests, e.g. originating from humans, but it is not allowed to merge them. This reflects established practices in science, where eventually a human is reponsible for data anaylsis code that becomes part of the project. Additional tasks git-bob is capable of are: 1) TThe assistant can support users of open source libraries by providing advice and code examples,as shown in Supplementary Figure S1. In case the assistant is not sure about the answer, it is capable of forwarding the question to a human, as shown in Supplementary Figure S2. 2) It can write and execute data analysis code, e.g. to summarize and plot CSV files which are stored in a repository, as demonstrated in Supplementary Figure S3. When analysing data, the assistant is intrinsically limited by the capabilities of the used LLM and on how detailed instructions are.

A highlight of git-bob is that no local installation or institutional internet server is required. Git-bob is implemented as GitHub Workflow, and hence runs within the IT infrastructure of github.com. It is compatible with OpenAI's LLMs such as GPT4, Anthropic's Claude, Google Gemini, and models hosted on Github Models Marketplace. Git-bob reports which model was used in all of its messages, as good scientific practice may suggest. The first three of these vendors require payment for using their services through an application programming interface (API), the fourth offers usage of their API for free. Obviously, the communication with the selected LLM is transmitted to the service provider, including source code files from the repository and images provided with the github issue. Hence, users are recommended to not submit any personal or sensitive information.

No new graphical interface needs to be learned, as git-bob integrates well with pre-existing workflows, formerly only between humans, now facilitating interaction with LLMs. Users also do not need to copy and paste code fragments between chat windows and code editors. Furthermore, unlike many LLM service providers, it allows interaction of multiple humans with the LLM within the same context. E.g. an open source software user can reach out with a question about how to analyse an image, a bio-image analysis expert can point out a potential strategy and the LLM implements the details, which can then be reviewed by the user and the expert. With this, multi-

* Corresponding Author. Email: robert.haase@uni-leipzig.de

Figure 1. example interaction with git-bob - figure placeholder

turn interaction with the LLM is enabled using direct text input and additional files from a given repository. When used with a vision language model (VLM) such as OpenAI's GPT4-omni, that can take an image as optional input, it can describe image content and with this, may have a better informed starting point for designing a bio-image analysis workflow. Git-bob can be configured for different purposes, such as assisting in code writing, data analysis, manuscript writing, code reviewing, and as outlined above for answering questions of externals reaching out to developers of open source repositories. This configuration is done by modifying system messages in the Github Workflow configuration file.

Using git-bob and also LLMs in general does not eliminate the need for expertise in the domain we are working in. For example, when tasked to setup a bio-image analysis workflow, at least one of the the humans interacting with git-bob should have the expertise to judge if a proposed solution makes sense. When LLMs improve, especially their reasoning capabilities, this limitation may feel less and less relevant. Eventually, as correct results are commonly unknown in data analysis research projects, a human may still be required who has the right expertise to judge if a workflow is reasonable.

Git-bob can be used in private repositories giving scientists the necessary privacy to work on code and documentation before they eventually publish their work. For example, this manuscript was edited with LLM-assistance in a private repository, and after making it public, any reader can finally see which modifications were done by the human, and how the AI-assistant contributed to the work as shown in in Figure 1C.

3 Conclusion

LLMs are being integrated in contemporary scientific workflows unavoidably, but documentation of how in detail they are imployed is commonly not done, also because of lack of tools doing this conveniently. To use LLMs responsibly, documenting how they were used in a specific project seems good-scientific-practice. If the scientific community documented usage of LLM prompts like they document usage of open source data analysis libraries, we could learn from each other how to prompt efficiently and responsibly. Git-bob allows facilitating this on multiple levels: for code generation, but also for manuscript writing. It works integrated on github.com enabling scientists to interact with an LLM via Github Issues and Pull-Requests offering new ways for establishing good scientific practice for the interaction between humans and artificial intelligence when working on projects together.

Acknowledgements

RH acknowledges the financial support by the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI.

References

[1] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

[2] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[3] L. A. Royer. The future of bioimage analysis: a dialog between mind and machine. *Nature Methods*, 20(7):951–952, 2023. ISSN 1548-7105. doi: 10.1038/s41592-023-01930-y. URL https://doi.org/10.1038/s41592-023-01930-y.

[4] L. A. Royer. Omega – harnessing the power of large language models for bioimage analysis. https://doi.org/10.5281/zenodo.8240289, 2023.

4 Supplementary material

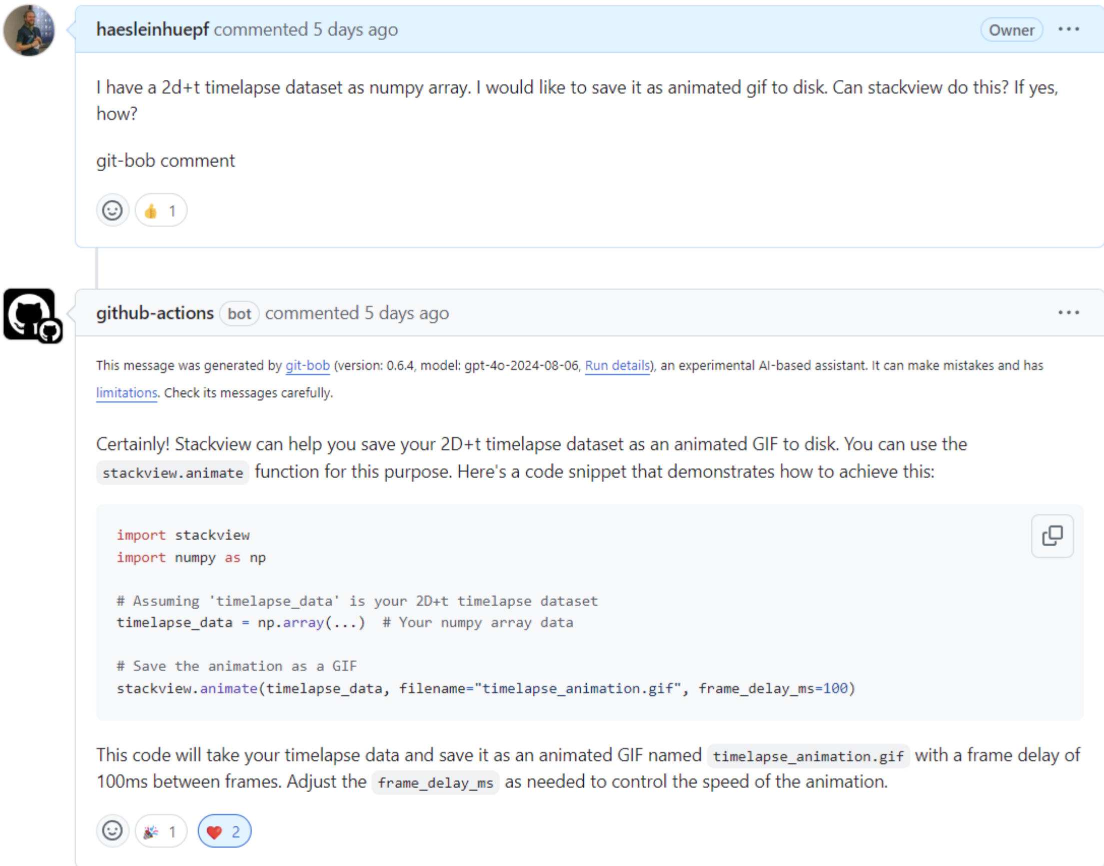

Figure S1. Use-case example for supporting users: The assistant can be configured to act as expert on a specific Python library and answer user questions. The entire discussion and corresponding code can be read online: https://github.com/haesleinhuepf/stackview/issues/79

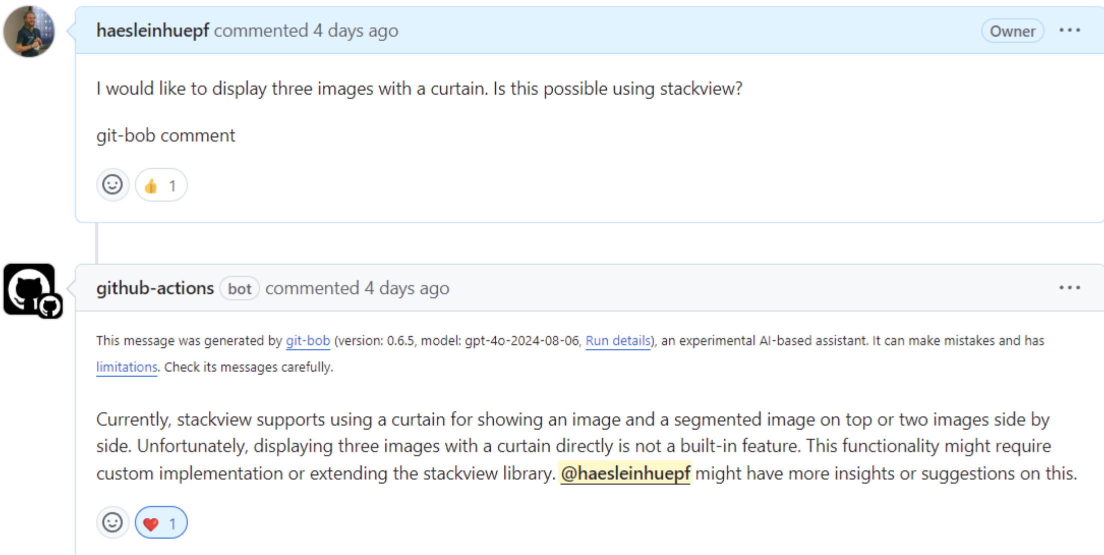

Figure S2. Use-case example for asking an expert: The answer to the question shown here is "No", but this is nowhere written in the documentation or the configuration of the assistant. In this case the assistant is not sure, and it can be configured to forward a question to a maintainer of the library where the question arrived. The entire discussion and corresponding code can be read online: https://github.com/haesleinhuepf/stackview/issues/80

Figure S3. Use-case example for plotting data: After explaining the assistant the folder structure of the project, it generates code for parsing a folder of CSV files and plotting results. The enter discussion and corresponding code can be read online: https://github.com/NFDI4BIOIMAGE/training/issues/250