Software Impacts 21 (2024) 100663

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Makita—A workflow generator for large-scale and reproducible simulation studies mimicking text labeling

Jelle Jasper Teijema [a,*], Rens van de Schoot [a], Gerbrich Ferdinands [a], Peter Lombaers [a], Jonathan de Bruin [b]

[a] Department of Methodology and Statistics, Faculty of Social and Behavioral Sciences, Utrecht University, Utrecht, The Netherlands
[b] Department of Research and Data Management Services, Information Technology Services, Utrecht University, Utrecht, The Netherlands

ARTICLE INFO

Keywords:
Systematic reviews
Machine learning
Simulation study
Reproducibility
Workflow automation

ABSTRACT

This paper introduces ASReview Makita, a tool designed to enhance the efficiency and reproducibility of simulation studies in systematic reviews. Makita streamlines the setup of large-scale simulation studies by automating workflow generation, repository preparation, and script execution. It employs Jinja and Python templates to create a structured, reproducible environment, aiding both novice and expert researchers. Makita's flexibility allows for customization to specific research needs, ensuring a repeatable research process. This tool represents an advancement in the field of systematic review automation, offering a practical solution to the challenges of managing complex simulation studies.

Code metadata

Current code version	V0.9.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-492
Permanent link to reproducible capsule	https://codeocean.com/capsule/5417339/tree/v2
Legal code license	MIT License
Code versioning system used	git
Software code languages, tools and services used	Python, Jinja
Compilation requirements, operating environments and dependencies	Programming Language :: Python :: 3.7 Programming Language :: Python :: 3.8 Programming Language :: Python :: 3.9 Programming Language :: Python :: 3.10 Programming Language :: Python :: 3.11 ASReview jinja2 cfgtemplater
If available, link to developer documentation/manual	https://github.com/asreview/asreview-makita/blob/main/README.md
Support email for questions	asreview@uu.nl

1. Summary

The field of accelerating the screening phase of systematic reviews with advanced machine learning methods is rapidly evolving [1]. A simulation study involves mimicking the screening process for a systematic review of a human in interaction with an Active learning model. The simulation reenacts the screening process as if a researcher were using a machine learning model to prioritize the order of papers being screened. The performance of one or multiple model(s) can then be measured by performance metrics, such as the Work Saved over Sampling, recall at a given point in the screening process, or the average time to discover a relevant record. However, setting up a simulation study can be a time-consuming and error-prone process, especially since reproducibility is of key importance.

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail address: j.j.teijema@uu.nl (J.J. Teijema).

https://doi.org/10.1016/j.simpa.2024.100663
Received 30 November 2023; Received in revised form 6 May 2024; Accepted 8 May 2024

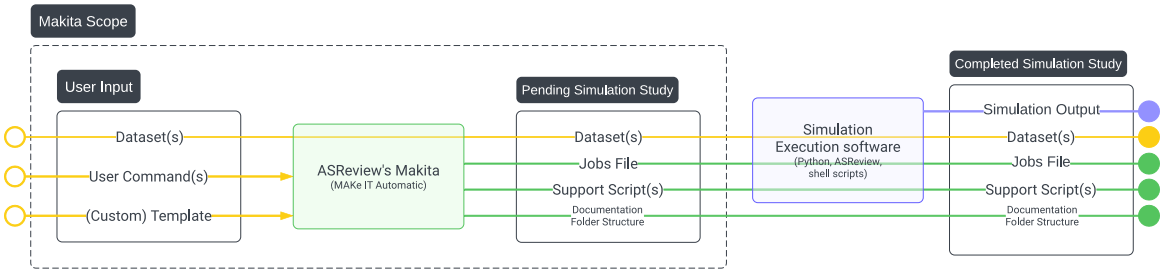

Fig. 1. Diagram illustrating the general workflow of Makita. The "Makita Scope" section defines the specific functionalities covered in this paper.

This paper presents **ASReview's Makita** (**MAKe IT A**utomatic) [2]. Makita is the precursor to a reproducible simulation study. It streamlines the simulation study design process for systematic reviews using ASReview [3], providing a generative framework to simplify creating and running large-scale simulations. Using Makita templates, different study workflows can be generated to fit the study needs. If a study requires a unique template, a custom template can be used. Its implementation through the command-line interface aims to make reproducible and repeatable research easy and efficient, to assist both novice and expert researchers.

2. Statement of need

Although tools such as ASReview LAB [4] offer various ways to simulate the screening process in systematic reviews via its user interface, there is a need for automation in setting up the research environment for large-scale simulation research. Setting up the structure of a simulation study manually is prone to mistakes and a tedious task, especially when the scale of the simulation increases. ASReview Makita fills this gap by automating the workflow setup, preparing GitHub repositories, documentation, pre/post-processing code, and generating execution scripts.

Simplifying reproducibility and maintaining an organized folder structure are key elements in scientific research. They ensure that experiments can be reliably repeated and built upon by other researchers. A well-organized directory makes it easier to understand the workflow and locate files, and contributes to the transparency and credibility of the study [5].

3. Technical functionality

Using a combination of Jinja-based templates and Python templates, ASReview Makita automatically generates a hierarchical folder structure, a README.md (including descriptions, instructions, file tree, and data statements), any additional code used for pre-and post-processing, and a batch or shell execution script. Makita offers code for, among others, extracting dataset statistics [6], extracting simulation performance metrics such as Time to Discovery [7], merging those metrics into easy-to-read tables, generating word clouds [8], and plotting the results [9]. Makita assures that all steps of the simulation study are stored and thus reproducible and transparent.

The Jinja-based templates handle study structure while accompanying Python templates add extended functionality. A range of standard templates is available, specifically tailored for ASReview simulations. Overall, the architecture provides a modular and flexible framework, allowing users to easily adapt the tool to their specific research needs.
What Makita does:
- Set up a workflow for running a large-scale simulation study
- Prepare a GitHub repository, including a readme file
- Automate the many lines of code needed
- Create an execution script for running the simulation study
- Make research fully reproducible
- Support custom templates for unique research questions

What Makita does not do:
- Execute jobs or tasks itself
- Write the study

While Makita was originally developed for use with ASReview's simulation CLI, Makita's design allows it to be integrated with any other CLI tool via a customized template, broadening its applicability across different large-scale research environments. Makita can be used locally, on a server, or can be used in combination with Docker and Kubernetes.

Very-large-scale simulation studies have been successfully run using Makita, with over 29.000 simulations in a single study, using 25 different datasets and 92 different simulation models (Teijema, J. J., [10]). The study implemented Makita within a Kubernetes cluster, generating custom templates on the fly for each of the cluster nodes' specific needs [11].

4. Software scope

The software discussed in this paper focuses on the elements in the section 'Makita Scope' of Fig. 1. For starting the creation of a simulation study, three elements are necessary: (1) one or more datasets that provide the input data, (2) user-defined commands that specify operational settings, and (3) a choice between using a default template or a custom template provided by the user. These elements are part of the 'User Input' section.

The following section is the 'Pending Simulation Study' section, which sets up but does not start the simulations. It organizes datasets, the jobs file, support scripts, documentation, and folder structure, preparing everything needed for the simulation study.

The 'Completed Simulation Study' section deals with the results after the simulations have run via execution of the jobs file. It includes output (such as, but not limited to: simulation files, plots, metric files, and updated documentation), input datasets, used code scripts for processing results, and documentation. Ideally, additional details should be added to the documentation to explain the goal of the study, results, and methods used, for future reference and reproducibility.

5. Software architecture

Makita starts with the `MakitaEntryPoint`, which handles the execution of user commands via the `argparse` library. The entry point defines several commands, with the primary ones being template setup and script addition (see Fig. 2).

For template configuration, the user specifies various operational parameters, such as the *template name*, *job file type*, *dataset and output locations*, *initialization seeds*, *model configurations*, and more, which allows for precise control over the simulation settings. The command setup supports dynamic handling of template parameters, allowing both default and user-provided templates. The template system uses Python's entry point mechanism to load templates, ensuring modularity and extensibility.

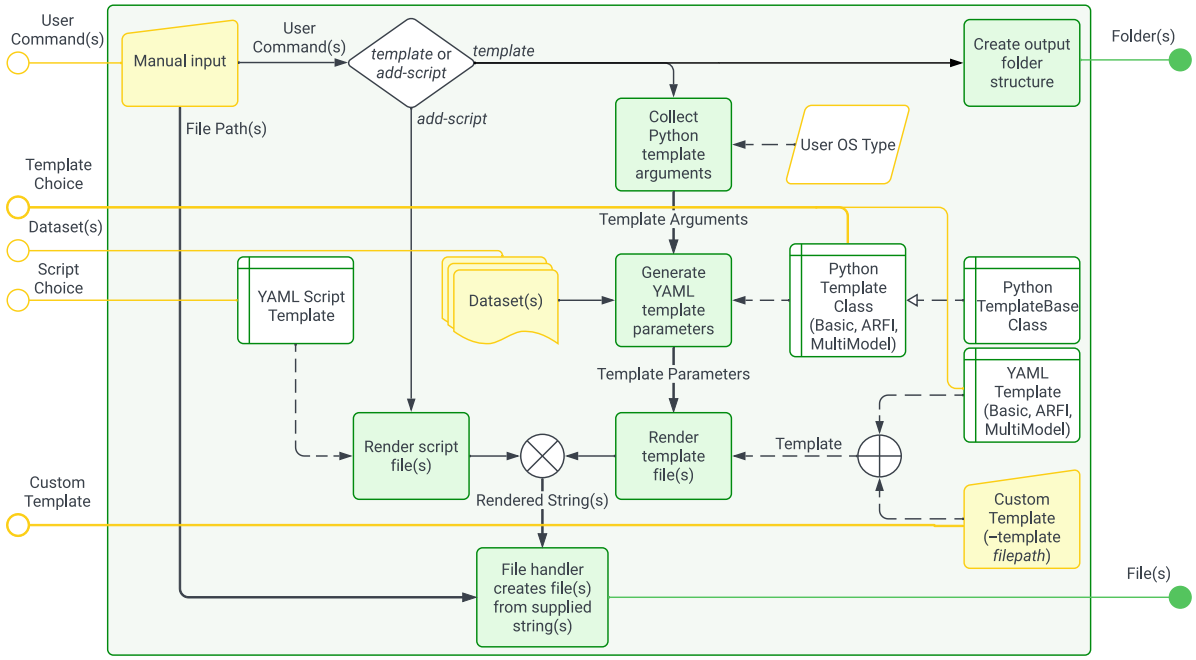

Fig. 2. Diagram illustrating the sequence of operations of Makita. Key processes include the selection of input datasets, choice of templates, collection of template arguments, and rendering of files from templates. Makita supports custom study design through user-defined templates and adapts the jobs file to the user's operating system. The final outputs are organized into a structured folder system as specified by the generated jobs file.

After the rendering of both scripts and documentation, files are centrally handled by the `FileHandler` class, which executes file operations such as adding, overwriting, and generating files from templates using the Jinja2 templating engine. The architecture is designed to be adaptive to different operating systems, providing tailored job file generation (e.g., *.bat* for Windows and *.sh* for Unix-like systems) across platforms.

6. Usage

```
asreview makita template basic _n_runs 100
```

Upon creating a '*data*' folder with the desired datasets, the study structure is generated by running the Makita command for the '*basic*' template. The Makita command can be customized with various flags to specify the study design. In this example, the '*n_runs*' flag is set to 100, indicating that 100 simulations are needed for the study. Executing the generated jobs file starts the simulations, producing simulation output, logs, and metrics. Fig. 3 shows the file tree results for running the basic template, and Fig. 4 after execution of the jobs file. File trees are generated with scientific ordering, following Scitree [12].

```
Makita_basic/
├── README.md
├── jobs.bat
├── data/
│   └── generic_labels.csv
└── scripts/
    ├── get_plot.py
    ├── merge_descriptives.py
    ├── merge_metrics.py
    └── merge_tds.py
```

Fig. 3. File tree structure generated by Makita for the 'basic'template, before execution of the jobs file. This structure is organized with scientific ordering.

```
└── output/
    ├── figures/
    │   ├── plot_recall_sim_generic_labels.png
    │   ├── wordcloud_generic_labels.png
    │   ├── wordcloud_irrelevant_generic_labels.png
    │   └── wordcloud_relevant_generic_labels.png
    ├── simulation/
    │   └── generic_labels/
    │       ├── descriptives/
    │       │   └── data_stats_generic_labels.json
    │       ├── metrics/
    │       │   ├── metrics_sim_generic_labels_0.json
    │       │   ├── ...
    │       │   └── metrics_sim_generic_labels_99.json
    │       └── state_files/
    │           ├── sim_generic_labels_0.asreview
    │           ├── ...
    │           └── sim_generic_labels_99.asreview
    └── tables/
        ├── metrics_sim_all.csv
        ├── metrics_sim_all.xlsx
        ├── data_descriptives_all.csv
        ├── data_descriptives_all.xlsx
        ├── metrics/
        │   ├── metrics_sim_generic_labels.csv
        │   └── metrics_sim_generic_labels.xlsx
        └── time_to_discovery/
            ├── tds_sim_generic_labels.csv
            └── tds_sim_generic_labels.xlsx
```

Fig. 4. File tree structure generated by Makita for the 'basic' template, after execution of the jobs file. This structure is organized with scientific ordering.

7. Impact overview

Enhanced Accessibility and Efficiency in Research: ASReview Makita significantly reduces the time and complexity involved in setting up simulation studies. Its ability to swiftly create reproducible workflows allows researchers, especially those new to the field, to initiate and evaluate their simulation studies in mere minutes. This accelerated process not only saves time but also encourages a broader

exploration of research questions, expanding the horizons of systematic review research.

Reproducibility and Reliability: The software's emphasis on reproducibility is a key element for improving the quality of research. By ensuring that each step of the simulation study is meticulously recorded and can be replicated, Makita enhances the reliability and credibility of research findings. On top of recording every step in Makita's process, it writes basic documentation for later reference. This documentation is crucial in a field where the accuracy and consistency of data processing directly influence the outcomes and interpretations of systematic reviews.

Ongoing Research and Contributions: The impact of Makita is evidenced by its usage in multiple research projects, both many unpublished exploratory studies and published works [13–17]. Makita is used in both government and commercial organizations, its open-source nature allowing for easy adaptation in any setting. These organizations include, but are not limited to, the *PBL Netherlands Environmental Assessment Agency*, the Dutch *National Institute for Public Health and the Environment*, and private institutions. Its usage in these projects highlights its utility and relevance in modern research settings.

CRediT authorship contribution statement

Jelle Jasper Teijema: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing, Validation. **Rens van de Schoot:** Conceptualization, Funding acquisition, Project administration, Supervision, Validation, Writing – review & editing. **Gerbrich Ferdinands:** Conceptualization, Software, Writing – review & editing. **Peter Lombaers:** Conceptualization, Software, Visualization. **Jonathan de Bruin:** Conceptualization, Data curation, Methodology, Software, Supervision, Validation, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J.J. Teijema, S. Seuren, D. Anadria, A. Bagheri, R. van de Schoot, Simulation based active learning for systematic reviews: A systematic review of the literature, 2023, http://dx.doi.org/10.31234/osf.io/67zmt, PsyArXiv.

[2] ASReview-LAB-developers, ASReview makita: A workflow generator for simulation studies using the command line interface of ASReview LAB, 2023, http://dx.doi.org/10.5281/zenodo.11103503.

[3] R. Van De Schoot, J. De Bruin, R. Schram, P. Zahedi, J. De Boer, F. Weijdema, B. Kramer, M. Huijts, M. Hoogerwerf, G. Ferdinands, et al., An open source machine learning framework for efficient and transparent systematic reviews, Nat. Mach. Intell. 3 (2) (2021) 125–133, http://dx.doi.org/10.1038/s42256-020-00287-7.

[4] ASReview-LAB-developers, ASReview LAB - A Tool for AI-Assisted Systematic Reviews, Zenodo, 2019, http://dx.doi.org/10.5281/zenodo.3345592.

[5] P. Lombaers, J. de Bruin, R. van de Schoot, Reproducibility and data storage for active learning-aided systematic reviews, Appl. Sci. 14 (9) (2024) 3842, http://dx.doi.org/10.3390/app14093842.

[6] ASReview-LAB-developers, ASReview datatools, 2022, http://dx.doi.org/10.5281/zenodo.7333281.

[7] G. Ferdinands, R. Schram, J. de Bruin, A. Bagheri, D.L. Oberski, L. Tummers, J.J. Teijema, R. van de Schoot, Performance of active learning models for screening prioritization in systematic reviews: A simulation study into the average time to discover relevant records, Syst. Rev. 12 (1) (2023) 100, http://dx.doi.org/10.1186/s13643-023-02257-7.

[8] ASReview-LAB-developers, ASReview wordcloud: A tool to create a visual impression of the verbal content within a systematic review dataset, 2022, http://dx.doi.org/10.5281/zenodo.6625855.

[9] ASReview-LAB-developers, ASReview insights, 2022, http://dx.doi.org/10.5281/zenodo.7418934.

[10] J.J. Teijema, J. de Bruin, A. Bagheri, R. van de Schoot, Large-scale simulation study of active learning models for systematic reviews, 2023, http://dx.doi.org/10.31234/osf.io/2w3rm.

[11] J.J. Teijema, Jteijema/asreview-simulation-project, 2023, http://dx.doi.org/10.5281/zenodo.7991081.

[12] J. De Bruin, Scitree - like tree, but optimized for science, 2023, http://dx.doi.org/10.5281/zenodo.7500084.

[13] D.G. Campos, T. Fütterer, T. Gfrörer, R.E. Lavelle-Hill, K. Murayama, L. König, M. Hecht, S. Zitzmann, R. Scherer, Screening smarter, not harder: A comparative analysis of machine learning screening algorithms and heuristic stopping criteria for systematic reviews in educational research, 2023, http://dx.doi.org/10.31234/osf.io/fpwc2.

[14] R.C. Neeleman, C. Leenaars, M. Oud, F. Weijdema, R. van de Schoot, Addressing the challenges of reconstructing systematic reviews datasets, 2023, http://dx.doi.org/10.31234/osf.io/jfcbq.

[15] M. Oude Wolcherink, X. Pouwels, S. van Dijk, C. Doggen, H. Koffijberg, Can artificial intelligence separate the wheat from the chaff in systematic reviews of health economic articles? Expert Rev. Pharm. Outcomes Res. (2023) 1–8, http://dx.doi.org/10.1080/14737167.2023.2234639.

[16] J.J. Teijema, L. Hofstee, M. Brouwer, J. de Bruin, G. Ferdinands, J. de Boer, P. Vizan, S. van den Brand, C. Bockting, R. van de Schoot, et al., Active learning-based systematic reviewing using switching classification models: The case of the onset, maintenance, and relapse of depressive disorders, Front. Res. Metr. Anal. 8 (2023) 1178181, http://dx.doi.org/10.3389/frma.2023.1178181.

[17] S. Romanov, A.S. Siqueira, J. de Bruin, J. Teijema, L. Hofstee, R. van de Schoot, Optimizing ASReview simulations: A generic multiprocessing solution for 'light-data' and 'heavy-data' users, Data Intell. (2024) 1–19, http://dx.doi.org/10.1162/dint_a_00244.