The Journal of Open Source Software

AutoRA: Automated Research Assistant for Closed-Loop Empirical Research

Sebastian Musslick [1,2¶], **Benjamin Andrew** [1], **Chad C. Williams** [1], **Joshua T. S. Hewson** [1], **Sida Li** [3], **Ioana Marinescu** [4], **Marina Dubova** [5], **George T. Dang** [1], **Younes Strittmatter** [1,4*], **and John G. Holland** [1*]

1 Brown University, USA **2** Osnabrück University, Germany **3** University of Chicago, USA **4** Princeton University, USA **5** University of Indiana, USA **¶** Corresponding author ***** These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Automated Research Assistant (`autora`) is a Python package for automating and integrating empirical research processes, such as experimental design, data collection, and model discovery. With this package, users can define an empirical research problem and specify the methods they want to employ for solving it. `autora` is designed as a declarative language in that it provides a vocabulary and set of abstractions to describe and execute scientific processes and to integrate them into a closed-loop system for scientific discovery. The package interfaces with other tools for automating scientific practices, such as `scikit-learn` for model discovery, `sweetpea` and `sweetbean` for experimental design, `firebase_admin` for executing web-based experiments, and `autodoc` for documenting the empirical research process. While initially developed for the behavioral sciences, `autora` is designed as a general framework for closed-loop scientific discovery, with applications in other empirical disciplines. Use cases of `autora` include the execution of closed-loop empirical studies (Musslick et al., 2024), the benchmarking of scientific discovery algorithms (Hewson et al., 2023; Weinhardt et al., 2024), and the implementation of metascientific studies (Musslick et al., 2023).

Statement of Need

The pace of empirical research is constrained by the rate at which scientists can alternate between the design and execution of experiments, on the one hand, and the derivation of scientific knowledge, on the other hand (Musslick et al., in press). However, attempts to increase this rate can compromise scientific rigor, leading to lower quality of formal modeling, insufficient documentation, and non-replicable findings. `autora` aims to surmount these limitations by formalizing the empirical research process and automating the generation, estimation, and empirical testing of scientific models. By providing a declarative language for empirical research, `autora` offers greater transparency and rigor in empirical research while accelerating scientific discovery. While existing scientific computing packages solve individual aspects of empirical research, there is no workflow mechanic for integrating them into a single pipeline, e.g., to enable closed-loop experiments. `autora` offers such a workflow mechanic, integrating Python packages for automating specific aspects of the empirical research process.

The Journal of Open Source Software

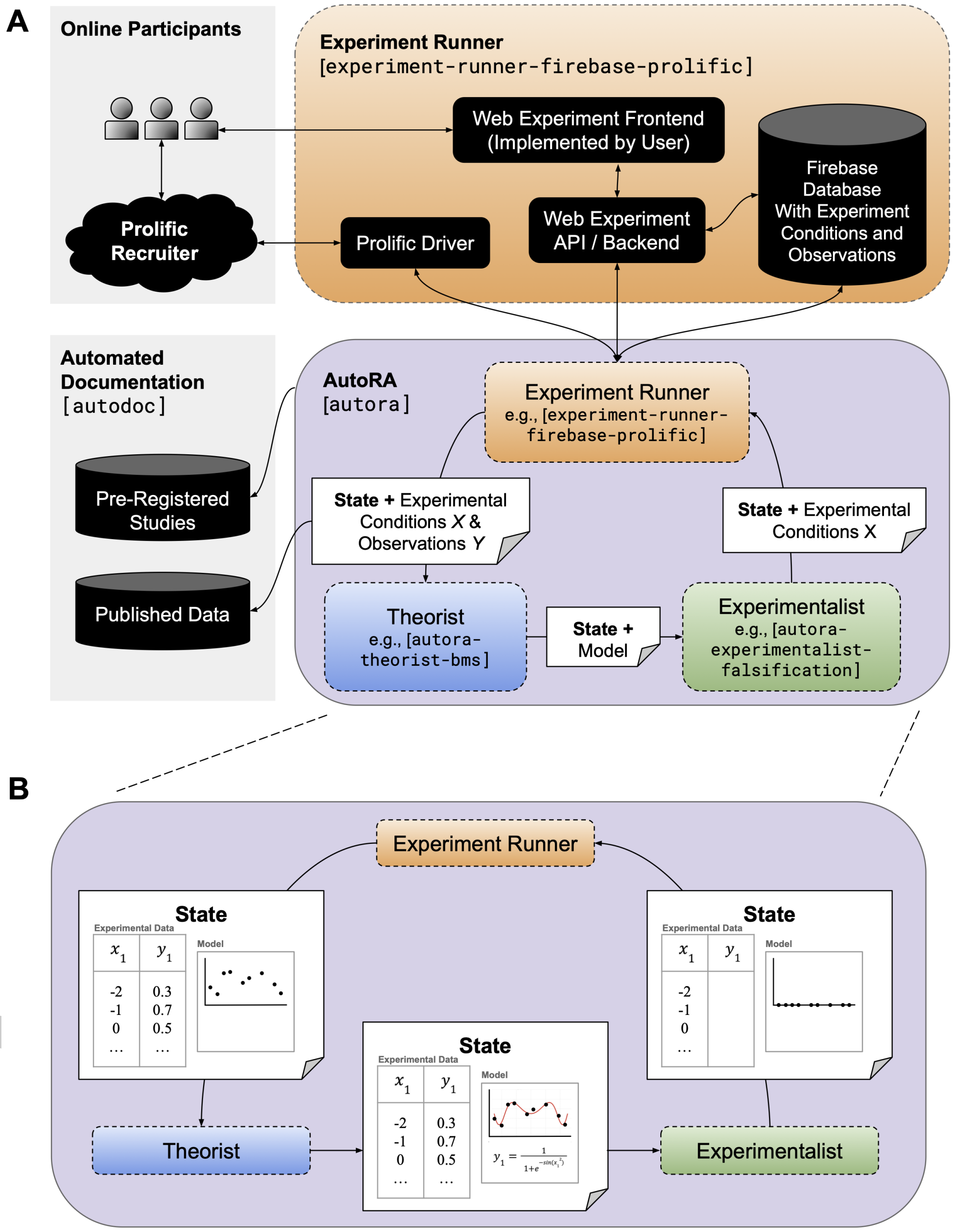

Figure 1: The autora framework illustrated for closed-loop behavioral research. (A) Exemplary autora workflow. autora implements components (colored boxes; see text) that can be integrated into a closed-loop discovery process. Workflows expressed in autora depend on modules for individual scientific tasks, such as designing behavioral experiments, executing those experiments, and analyzing collected data. (B) autora's components acting on the state object. The state object maintains relevant scientific data, such as experimental conditions X, observations Y, and models, and can be modified by autora components. Here, the cycle begins with an experimentalist adding experimental conditions x_1 to the state. The experiment runner then executes the experiment and collects corresponding observations y_1. The cycle concludes with the theorist computing a model that relates x_1 to y_1.

Overview and Components

The `autora` framework implements and interfaces with components automating different phases of the empirical research process (Figure 1A). These components include *experimentalists* for automating experimental design, *experiment runners* for automating data collection, and *theorists* for automating scientific model discovery. To illustrate each component, we consider an exemplary behavioral research study (cf. Figure 1) that examines the probability of human participants detecting a visual stimulus as a function of its luminosity.

Experimentalist components take the role of a research design expert, determining the next iteration of experiments to be conducted. Experimentalists are functions that identify experimental conditions which can be subjected to measurement by experiment runners, such as different levels of stimulus luminosity. To determine these conditions, experimentalists may use information about candidate models obtained from theorist components, experimental conditions that have already been probed, or respective observations. The `autora` framework offers various experimentalist packages, each for determining new conditions based on, for example, novelty, prediction uncertainty, or model disagreement (Dubova et al., 2022; Musslick et al., 2023).

Experiment runner components correspond to research technicians collecting data from an experiment. They are implemented as functions that accept experimental conditions as input (e.g., a pandas dataframe with columns representing different experimental variables) and produce collected observations as output (e.g., a pandas dataframe with columns representing different experimental variables along with corresponding measurements). `autora` (4.2.0) provides experiment runners for two types of automated data collection: real-world and synthetic. Real-world experiment runners include interfaces for collecting data in the real world. For example, the `autora` framework offers experiment runners for automating the data collection from web-based experiments for behavioral research studies (Musslick et al., 2024). In the behavioral experiment described above, an experiment runner may set up a web-based experiment that measures the probability of human participants detecting visual stimuli with varying luminosities. These runners interface with external components including recruitment platforms (e.g, Prolific; Palan & Schitter (2018)) for coordinating the recruitment of participants, databases (e.g., Google Firestore) for storing collected observations, and web servers for hosting the experiments (e.g., Google Firebase). Synthetic experiment runners act as simulators for real-world experiments: they specify the data-generating process and collect observations from it. For example, `autora-synthetic` implements established models of human information processing (e.g, for perceptual discrimination) and conducts experiments on them. These synthetic experiments serve multiple purposes, such as testing and benchmarking `autora` components before applying them in the real-world (Musslick et al., 2024) or conducting computational metascience studies (Musslick et al., 2023).

Theorist components embody the role of a computational scientist, employing modeling techniques to find a model that best characterizes, predicts, and/or explains the study's observations. Theorists may identify different types of scientific models (e.g., statistical, mathematical, or computational) implemented as `scikit-learn` estimators (Pedregosa et al., 2011). In case of the behavioral research study, a model may correspond to a psychophysical law relating stimulus luminosity to the probability of detecting the stimulus. `autora` provides interfaces for various equation discovery methods that are implemented as `scikit-learn` estimators, such as deep symbolic regression (Landajuela et al., 2022; Petersen et al., 2021), `PySR` (Cranmer et al., 2020), and the Bayesian Machine Scientist (Guimerà et al., 2020; Hewson et al., 2023). Alternatively, a model may correspond to a fine-tuned large language model (Binz et al., 2024), enabling its automated alignment with human behavior from web-based experiments. A model is generated by fitting experimental data. Accordingly, theorists take as input a pandas dataframe specifying experimental conditions (instances of experimental variables) along with corresponding observations to fit a respective model. The model can then be used to generate predictions, e.g., to inform the design of a subsequent experiment.

The Journal of Open Source Software

Design Principles and Packaging

autora was designed as a general framework aimed at democratizing the automation of empirical research across the scientific community. Key design decisions were: 1) using a functional paradigm for the components and 2) splitting components across Python namespace packages.

Each component is a function that operates on immutable "state objects" which represent data from an experiment (Figure 1B), such as proposed experimental conditions, corresponding observations (represented as a pandas dataframe), and scientific models (represented as a list of `scikit-learn` estimators). Data produced by each component can be seen as additions to the existing data stored in the state. Thus, each component C takes in existing data in a state S, adds new data ΔS, and returns an updated state S',

$$S' = C(S) = S + \Delta S.$$

Accordingly, the components share their interface – every component loads data from and saves data to state objects, so they can be ordered arbitrarily, and adding a new component is as simple as implementing a new function or `scikit-learn`-compatible estimator and wrapping it with a utility function provided in `autora-core`. State immutability allows for easy parallelism and reproducibility (so long as the components themselves have no hidden state).

The `autora` framework presumes that each component is distributed as a separate package but in a shared namespace, and that `autora-core` – which provides the state – has very few dependencies of its own. For users, separate packages minimize the time and storage required for an install of an `autora` project. For contributors, they reduce incidence of dependency conflicts (a common problem for projects with many dependencies) by reducing the likelihood that the library they need has an existing conflict in `autora`. It also allows contributors to independently develop and maintain modules, fostering ownership of and responsibility for their contributions. External contributors can request to have packages vetted and included as an optional dependency in the `autora` package.

Acknowledgements

The AutoRA framework is developed and maintained by members of the Autonomous Empirical Research Group. S. M., B. A., C. C. W., J. T. S. H., and Y. S. were supported by the Carney BRAINSTORM program at Brown University. S. M. also received support from Schmidt Science Fellows, in partnership with the Rhodes Trust. The development of auxiliary packages for AutoRA, such as `autodoc` or `autora-experiment-server`, is supported by the Virtual Institute for Scientific Software (VISS) as part of Schmidt Sciences, LLC. The AutoRA package was developed using computational resources and services at the Center for Computation and Visualization, Brown University.

References

Binz, M., Akata, E., Bethge, M., Brändle, F., Callaway, F., Coda-Forno, J., Dayan, P., Demircan, C., Eckstein, M. K., Éltető, N., & others. (2024). Centaur: A foundation model of human cognition. *arXiv Preprint arXiv:2410.20268*. https://doi.org/10.48550/arXiv.2410.20268

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., & Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, *33*, 17429–17442. https://doi.org/10.48550/arXiv.2006.11287

The Journal of Open Source Software

Dubova, M., Moskvichev, A., & Zollman, K. (2022). Against theory-motivated experimentation in science. *MetaArXiv*, *24*. https://doi.org/10.31222/osf.io/ysv2u

Guimerà, R., Reichardt, I., Aguilar-Mogas, A., Massucci, F. A., Miranda, M., Pallarès, J., & Sales-Pardo, M. (2020). A Bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, *6*(5), eaav6971. https://doi.org/10.1126/sciadv.aav6971

Hewson, J., Strittmatter, Y., Marinescu, I., Williams, C., & Musslick, S. (2023). Bayesian machine scientist for model discovery in psychology. *NeurIPS 2023 AI for Science Workshop*. https://openreview.net/forum?id=XHFfvzIQ1n

Landajuela, M., Lee, C. S., Yang, J., Glatt, R., Santiago, C. P., Aravena, I., Mundhenk, T., Mulcahy, G., & Petersen, B. K. (2022). A unified framework for deep symbolic regression. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35, pp. 33985–33998). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2022/file/dbca58f35bddc6e4003b2dd80e42f838-Paper-Conference.pdf

Musslick, S., Bartlett, L. K., Chandramouli, S. H., Dubova, M., Gobet, F., Griffiths, T. L., Hullman, J., King, R. D., Kutz, J. N., Lucas, C. G., Mahesh, S., Pestilli, F., Sloman, S. J., & Holmes, W. R. (in press). Automating the practice of science–opportunities, challenges, and implications. *Proceedings of the National Academy of Sciences*.

Musslick, S., Hewson, J. T., Andrew, B. W., Strittmatter, Y., Williams, C. C., Dang, G. T., Dubova, M., & Holland, J. G. (2023). *An evaluation of experimental sampling strategies for autonomous empirical research in cognitive science*. *45*, 1386–1392.

Musslick, S., Strittmatter, Y., & Dubova, M. (2024). *Closed-loop scientific discovery in the behavioral sciences*. https://doi.org/10.31234/osf.io/c2ytb

Palan, S., & Schitter, C. (2018). Prolific. Ac—A subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, *17*, 22–27. https://doi.org/10.1016/j.jbef.2017.12.004

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petersen, B. K., Larma, M. L., Mundhenk, T. N., Santiago, C. P., Kim, S. K., & Kim, J. T. (2021). Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1912.04871

Weinhardt, D., Eckstein, M. K., & Musslick, S. (2024). Computational discovery of human reinforcement learning dynamics from choice behavior. *NeurIPS 2024 Workshop on Behavioral Machine Learning*. https://openreview.net/forum?id=x2WDZrpgmB