The Journal of Open Source Software

meow: A Unified Framework for Conducting Simulations of Computer Adaptive Testing Algorithms in R

Klint Kanopka[1¶] **and Sophia Deng**[1]

1 New York University, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Computer adaptive testing (CAT) is an approach to assessment where the responses an examinee have previously given to questions play some role in deciding the subsequent questions they will be presented with. This style of testing allows for increased measurement precision with fewer items, a benefit for test administrators and test takers alike. Advances in CAT are done at the algorithmic level; new methods are developed to better select next items, update person and item parameters, control how often items are exposed to respondents, and decide when to stop administering new questions. Determining the potential measurement consequences of new algorithmic decisions requires developers to run comparison simulations that demonstrate effectiveness and tradeoffs. This demands porting the methods of other researchers into ad hoc simulation frameworks that are often designed to conduct single simulation studies. With `meow`, we modularize the core components of the CAT administration process and allow users to conduct standardized, comparable, and reproducible simulation studies.

Statement of need

`meow` is a package written in R to facilitate psychometric research in the computer adaptive testing (CAT) space that focuses on the development of new CAT algorithms. Software to simulate CAT data already exists, the most popular of which are `mirtCAT` (Chalmers, 2016) and `catR`(Magis & Barrada, 2017), but others also exist(Choi, 2009; Han, 2012; Oppl et al., 2017). The key to understanding the purpose of these packages (and the gap that `meow` fills) is that existing software facilitates simulating the *administration* of CATs for the purpose of comparing test designs in a given context with a given item pool. As such, these packages come with a selection of in-built item response theory models for ability estimation and pre-built item selection methods, exposure controls, and stopping rules. The issue, however, is that if a researcher is *developing* any new parameter update algorithms, item selection algorithms, exposure controls, or stopping rules, this requires a largely from-scratch implementation to conduct simulation studies that compare their new methods to existing approaches. As CAT research increasingly augments traditional psychometrics with computational approaches (Liu et al., 2024), the need for algorithmic comparison studies only increases. Using `meow`, users can easily implement new CAT algorithms that can be quickly integrated into existing simulation studies and shared with other researchers doing CAT development work.

Features

To avoid frequent reinvention of the wheel, `meow` modularizes CAT simulations by dividing the CAT administration into three distinct parts: (1) data generation, (1) item selection (including exposure control and a stopping rule), and (3) parameter updates. Each of these modules is supported by a central simulation framework, providing a consistent API to facilitate simple and reproducible simulation studies. While meow contains a selection of built-in data generating processes (DGPs), item selection algorithms, and parameter update methods, the advance is a documented and flexible platform in which users can implement their own versions of any of these pieces. The end result is the ability to swap algorithms in and out of larger simulation studies to provide directly comparable and reproducible results.

Simulations in `meow` are built around conducting a single call to the function `meow()`. As arguments, this takes an item selection function, a parameter update function, and a data loader function. These three components dictate how the simulation will be carried out, and `meow` comes bundled with off-the-shelf implementations of a selection of common parameter update and item selection algorithms, including classic methods and example implementations of recently published methods [Gorney & Reckase (2025);klinkenberg2011mathsgarden;Van der Linden et al. (2000);Vermeiren et al. (2025)]. Users also have the ability to supply initial values for internal parameters and set random seeds for each individual component of the simulation, allowing for customizability, reproducibility, and comparability between runs. Each simulation outputs consistently formatted estimates and bias for each internal parameter at each iteration of the simulation, allowing users to easily parse and recycle visualization code using commonly available `R` tools.

The real value of `meow` is to the developer of new algorithms, however. Using a common API, users can implement their own methodological developments quickly and easily in `meow`. Vignettes walk users through the development process. To facilitate easier development, internally parameter values are currently stored in dataframes. This has two advantages: First, it allows users to easily add more person or item parameters while still handing off the same internal objects. Second, it allows the use of `tidyverse` style data manipulation within the user-developed modules. This does have performance ramifications, however. As such, future versions of `meow` may implement changes to the structure of these objects in response to community feedback.

Output is designed to be easily visualized using commonly used tools, such as `ggplot2`. The output object structure is defined by the framework itself, not by any specific algorithmic component, allowing for reuse of visualization and analysis pipelines. A visualization-focused vignette walks users through understanding the structure of output objects and builds some commonly used visualizations, such as parameter trajectories and RMSE curves. Additionally, the output objects also contain adjacency matrices that count the number of respondents each pair of item has been exposed to. In addition to being useful for implementing simple exposure controls, this allows for simplified analysis and visualization of item utilization patterns.

The goal of the `meow` project is to advance CAT algorithm development research by serving as a central repository for implementations of new work. Users are encouraged to submit pull requests to merge their algorithmic advances back into the package so that they can easily have their work integrated into simulation studies conducted by future researchers. This goal has led to `meow` being developed with expandability and future maintenance in mind.

Ongoing Research

Development of `meow` was motivated by ongoing research by the authors in CAT algorithm development that models the item pool as a network object for the purposes of item selection, exposure control, and propagating parameter updates.

Acknowledgements

We specifically thank Kylie Gorney for valuable feedback on the implementation and usability of this software.

References

Chalmers, R. P. (2016). Generating adaptive and non-adaptive test interfaces for multidimensional item response theory applications. *Journal of Statistical Software*, *71*, 1–38. https://doi.org/10.18637/jss.v071.i05

Choi, S. W. (2009). Firestar: Computerized adaptive testing simulation program for polytomous item response theory models. *Applied Psychological Measurement*, *33*(8), 644. https://doi.org/10.1177/0146621608329892

Gorney, K., & Reckase, M. D. (2025). Using multiple maximum exposure rates in computerized adaptive testing. *Journal of Educational Measurement*. https://doi.org/10.1111/jedm.12436

Han, K. T. (2012). SimulCAT: Windows software for simulating computerized adaptive test administration. *Applied Psychological Measurement*, *36*(1), 64–66. https://doi.org/10.1177/0146621611414407

Liu, Q., Zhuang, Y., Bi, H., Huang, Z., Huang, W., Li, J., Yu, J., Liu, Z., Hu, Z., Hong, Y., & others. (2024). Survey of computerized adaptive testing: A machine learning perspective. *arXiv Preprint arXiv:2404.00712*. https://doi.org/10.48550/arXiv.2404.00712

Magis, D., & Barrada, J. R. (2017). Computerized adaptive testing with r: Recent updates of the package catR. *Journal of Statistical Software*, *76*, 1–19. https://doi.org/10.18637/jss.v076.c01

Oppl, S., Reisinger, F., Eckmaier, A., & Helm, C. (2017). A flexible online platform for computerized adaptive testing. *International Journal of Educational Technology in Higher Education*, *14*(1), 2. https://doi.org/10.1186/s41239-017-0039-0

Van der Linden, W. J., Glas, C. A., & others. (2000). *Computerized adaptive testing: Theory and practice* (Vol. 13). Springer.

Vermeiren, H., Kruis, J., Bolsinova, M., Maas, H. L. van der, & Hofman, A. D. (2025). Psychometrics of an elo-based large-scale online learning system. *Computers and Education: Artificial Intelligence*, *8*, 100376. https://doi.org/10.1016/j.caeai.2025.100376