Towards Accessible, Flexible and Collaborative Scientific Modeling with Terarium

PASCALE PROULX*†, JAMIE WAESE†, MENG-WEI CHANG†, LAI CHUNG LIU†, HOLLAND VASQUEZ†, NEIL GRAHAM†, DAVID GAULDIE, YOHANN PARIS, DEREK VINCE, CHARLES COLEMAN, KEVIN BIRK, JAEHWAN RYU, COLE BLANCHARD, EDWIN LAI, TOM SZENDREY, JULIAN WHITING, SHAWN YAMA, DAVID SCHROH, and DAVID JONKER, Uncharted Software, CA
BRANDON ROSE, MATTHEW PRINTZ, and FIVE GRANT, Jataware Corp., US

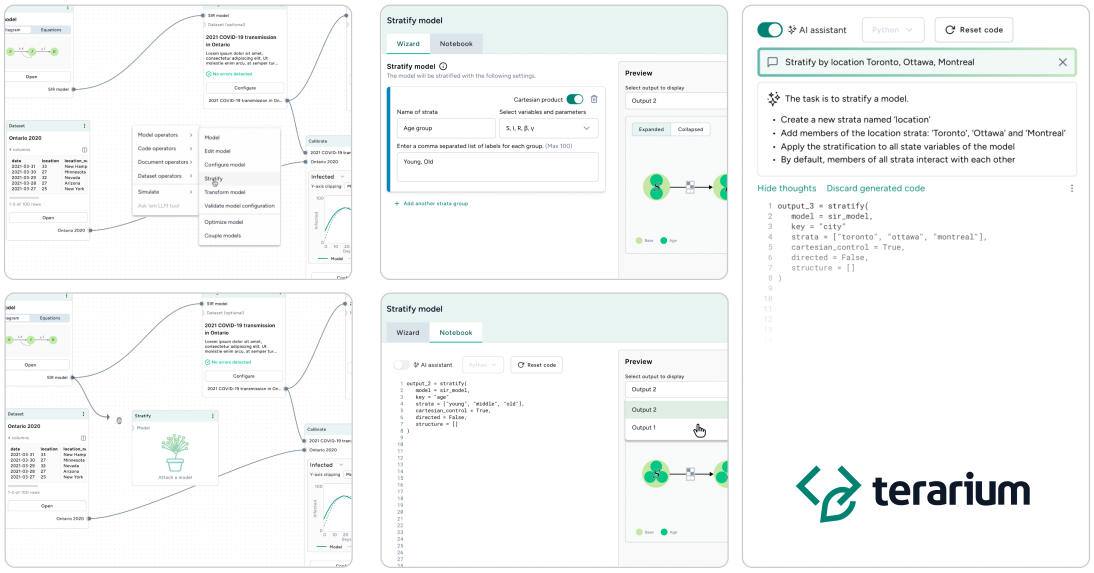

Fig. 1. Terarium's approach supports nonlinear workflows with computational notebooks and caters to varying expertise levels via dual-modality views and AI assistance. A: An asynchronous-executable graph of diverse code notebooks, adaptable for various scientific workflows. B: Users can toggle between wizard and notebook views, both of which offer output visualization for quick iterations and comparisons, providing the ability to learn from the code.

*Corresponding author
†These authors prepared this manuscript.

Authors' addresses: Pascale Proulx, pproulx@uncharted.software; Jamie Waese, jwaese@uncharted.software; Meng-Wei Chang, dchang@uncharted.software; Lai Chung Liu, nliu@uncharted.software; Holland Vasquez, hvasquez@uncharted.software; Neil Graham, ngraham@uncharted.software; David Gauldie, dgauldie@uncharted.software; Yohann Paris, yparis@uncharted.software; Derek Vince, dvince@uncharted.software; Charles Coleman, ccoleman@uncharted.software; Kevin Birk, kbirk@uncharted.software; Jaehwan Ryu, jryu@uncharted.software; Cole Blanchard, cblanchard@uncharted.software; Edwin Lai, elai@uncharted.software; Tom Szendrey, tszendrey@uncharted.software; Julian Whiting, jwhiting@uncharted.software; Shawn Yama, syama@uncharted.software; David Schroh, dschroh@uncharted.software; David Jonker, djonker@uncharted.software, Uncharted Software, 2 Berkeley St, Suite 600, Toronto, Ontario, CA, M5A 4J5; Brandon Rose, brandon@jataware.com; Matthew Printz, matt@jataware.com; Five Grant, five@jataware.com, Jataware Corp., 4707 Connecticut Ave NW, Apt 201, Washington, DC, US, 20008.

In the evolving field of scientific modeling research, computational notebooks like Jupyter [7] and Mathematica [13] have made steps to promote collaboration by providing interactive environments for code execution, visualization, and narrative composition; however they are still limited. For instance issues exist in supporting nonlinear workflows, scalability, and reproducibility leading to significant barriers in reusing work. To overcome these limitations, we introduce Terarium, a platform that attempts to socialize scientific modeling for a broad audience, including students, researchers, and data scientists. Terarium is designed to support nonlinear workflows, cater to varying expertise levels, and promote collaboration. Its workflow editor enables users to connect notebooks focused on specific modeling and simulation tasks, along with modeling assets, in flexible, branching workflows. Users can utilize dual-modality workflow nodes — a wizard and notebook view — to perform core operations with potentially less effort and gain insights into how to use the underlying code libraries to go further. Additionally, the AI-assistance should support users with a range of programming experience. By focusing on accessibility, flexibility, and code transparency, Terarium aims to address current challenges and foster a collaborative scientific modeling environment, enriching the dialogue on computational modeling tool development.

CCS Concepts: • **Human-centered computing** → **Interactive systems and tools**; • **Software and its engineering** → *Collaboration in software development*.

Additional Key Words and Phrases: Computation notebooks, scientific modeling, nonlinear workflows, collaboration, user interface design, AI-assisted analysis, accessibility

ACM Reference Format:
Pascale Proulx, Jamie Waese, Meng-Wei Chang, Lai Chung Liu, Holland Vasquez, Neil Graham, David Gauldie, Yohann Paris, Derek Vince, Charles Coleman, Kevin Birk, Jaehwan Ryu, Cole Blanchard, Edwin Lai, Tom Szendrey, Julian Whiting, Shawn Yama, David Schroh, David Jonker, Brandon Rose, Matthew Printz, and Five Grant. 2024. Towards Accessible, Flexible and Collaborative Scientific Modeling with Terarium. 1, 1 (March 2024), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Scientific modeling using computational notebooks, such as Jupyter [7] and Mathematica [13], has transitioned from simple interactive coding environments into sophisticated scientific research platforms that integrate code, visualizations, and narrative text. They have become indispensable for their ability to streamline the exploration, documentation, and sharing of reproducible computational analyses and results. While computational notebooks are powerful tools for scientific inquiry, they are not without constraints. There are limitations associated with linearity, provenance, and scalability. Furthermore, users face challenges related to version control, the scalability of data processing, and the replication of computational environments for consistent results across diverse setups.

We introduce Terarium, a modeling and simulation platform designed to make the intricate process of scientific modeling and scenario testing more accessible to a diverse range of users. By catering to researchers, graduate students, policy analysts, and decision makers, Terarium seeks to bridge the gaps that prevent broader engagement in scientific modeling exploration and analysis, thereby democratizing access to complex computational tools and methodologies.

Within this workshop we discuss how the Terarium platform attempts to address issues associated with computational notebooks. Our key contributions include: 1) A workflow editor that chains notebooks together into nonlinear workflows (Figure 1A), 2) dual-modality workflow operators (i.e., a linked graphical human-machine interface and an AI-assisted notebook) that accommodates different user expertise levels (Figure 1B), 3) methods for immediate visualization and

iteration alongside code that enhance comparisons and provenance and 4) support of multiple programming languages within a single workflow.

2 PRIOR ART

The landscape of computational notebooks has evolved significantly over the years, with Jupyter notebooks [7] emerging as a cornerstone in data science and research. These platforms have democratized access to computational tools, enabling researchers, educators, and practitioners to blend code, visualization, and narrative in a single interactive document. Jupyter notebooks, specifically, have been highlighted for their role in open science and their ability to combine executable code with output and notes, making them popular in various scientific fields for rapid experimentation and sharing results [2, 9]. However, they are typically bound to a single programming language per notebook, which can limit cross-language data analysis and tool integration, and their linear narrative structure impedes revisiting and modifying earlier parts of the analysis, especially in complex projects [1].

Graphical computational tools such as Stella [6] have sought to lower the barrier to entry even further, offering a more intuitive, visual approach to data analysis and model building. By abstracting the complexity of code, these tools aim to make computational analysis more accessible to a broader audience, including those with limited programming experience. Similarly, Einblick [5] leverages visual programming and collaborative features to make data science more approachable to a wider audience. These platforms underscore the industry's shift towards tools that not only simplify the data science process but also enhance real-time collaboration among researchers. However, this approach often constrains flexibility and depth. Graphical interfaces may limit users to predefined functionalities and workflows, making it difficult to execute tasks that fall outside the scope of the tool's designed use cases [8]. This trade-off highlights a gap in the current ecosystem of computational tools—a need for platforms that combine the accessibility of graphical interfaces with the flexibility and power of traditional coding environments. Bridging this gap could significantly enhance the capabilities of researchers and analysts, enabling more dynamic, interdisciplinary, and nuanced explorations of data.

Efforts to address these limitations have led to the development of innovative platforms like Lodestar, which supports rapid prototyping of data science workflows through data-driven analysis recommendations [8]. Furthermore, the introduction of "2D computational notebooks" offers a promising direction for overcoming the limitations of traditional Jupyter notebooks, providing enhanced efficiency and usability through a multi-column 2D computational space [3, 4, 11, 12]. Such advancements underscore the ongoing evolution of computational tools, striving for a balance between accessibility for novices and the demanding flexibility required by advanced users.

3 THE TERARIUM PLATFORM

3.1 Nonlinear Workflows

The Workflow Editor of Terarium (Figure 2) is an interactive interface for managing, integrating, and assembling scientific project resources, such as datasets and computational models, into a cohesive network. This network is structured around "operators," each of which is a self-contained programming environment. Each operator receives input resources, executes specified actions using a suitable programming language, and generates new resources that can be utilized by subsequent operators in the workflow (Figure 2A). This arrangement forms a directed multigraph, where diverse types of code notebooks are interconnected and can run asynchronously. The workflow nodes also provide a visual summary of the status and output of the operator (Figure 2B). With a growing library of operators, the

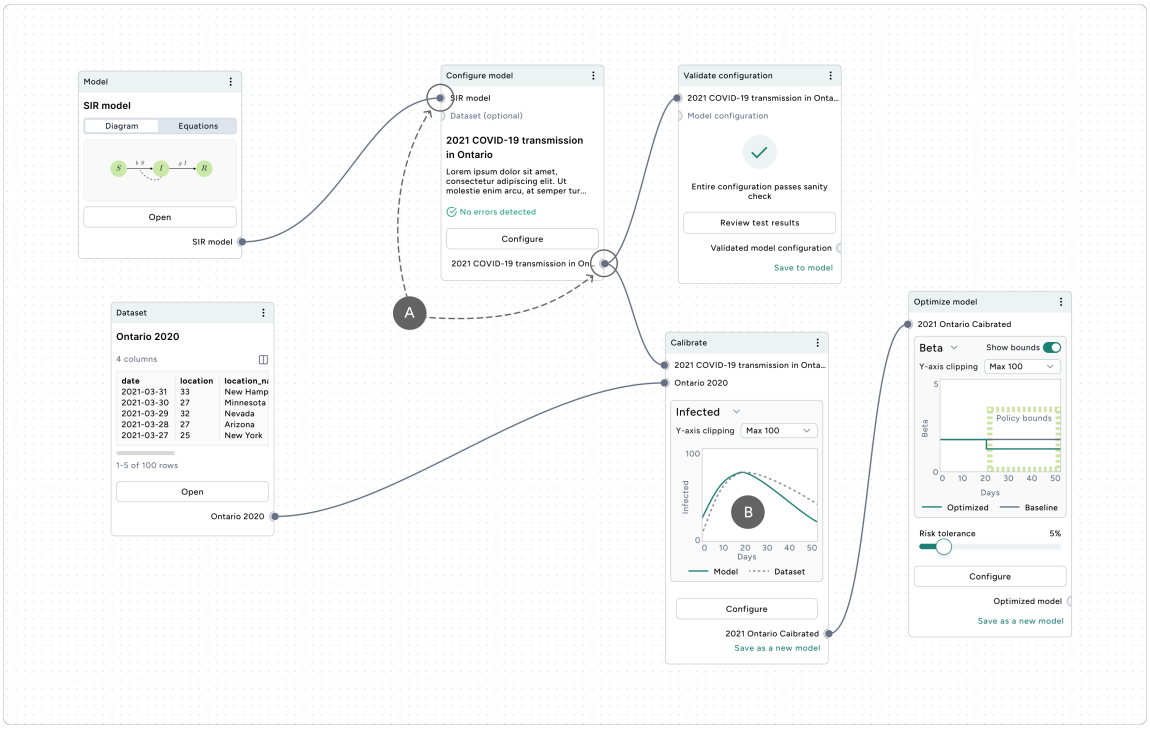

Fig. 2. Terarium Workflow Editor. A. Each operator takes input resources, executes actions with a suitable programming language, and generates new resources for subsequent operators. For example, the 'Configure model' operator takes a model and an optional dataset as inputs and outputs a model configuration, which in turn, can be used as an input for a 'Validate configuration' or a 'Calibrate' operator. B. The workflow nodes also visually summarize the asset or the operator's status and output.

Workflow Editor accommodates a variety of scientific workflows, catering to disciplines as varied as epidemiology for infectious disease forecasting and climatology for model intercomparison. The graphical interface of the Workflow Editor—based on simple drag-and-drop interactions—is designed to allow scientists to concentrate on their research without needing to understand how to orchestrate complex computational processes.

3.2 Dual-Modality Workflow Operators

Within the Editor, users are afforded a granular level of control over the configuration of each "workflow operator". A workflow operator is a computational notebook focused on a particular step in the modeling and simulation. A Wizard view designed for ease of use exposes each operator's configuration and suggested default values to streamline the setup process (Figure 3). This approach is designed to help users who prefer a more guided experience or those who do not require deep customization of their workflow components.

Conversely, a Notebook view that resembles an IDE-lite interface offers a more technical and flexible environment. In this view, users have an interactive programming session linked to the operator's underlying code. A code window pre-populated with any source code executed by the Wizard allows for direct editing and customization (Figure 4A). The Notebook view facilitates a higher degree of workflow customization, reproducibility, and transparency while also providing an opportunity to debug and learn from the workflow construction process. This dual-view setup

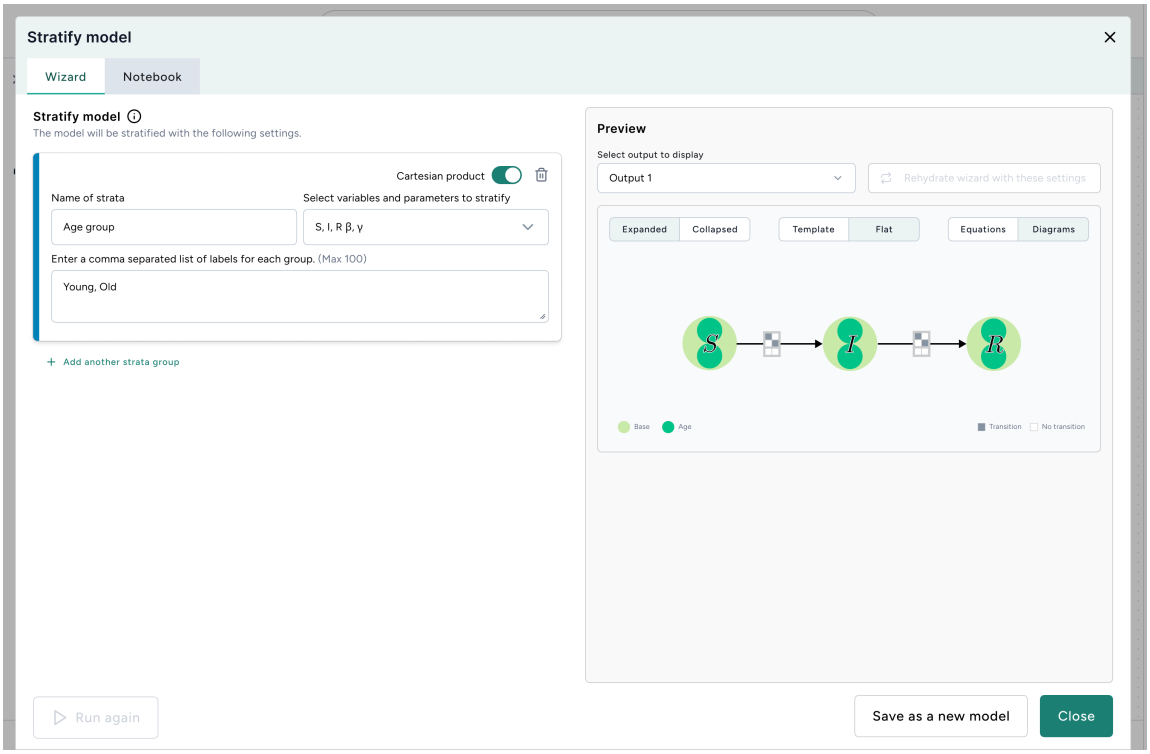

Fig. 3. Wizard view of the 'Stratify' operator. The wizard view of Terarium operators aims to optimize ease of use and generate quick results, leveraging the affordances of forms with example values.

enables users to tailor their workflows with precision, moving seamlessly between guided configuration and in-depth programming.

3.3 Immediate Visualization and Rapid Iteration

Terarium automatically stores results generated from the execution of each operator configuration. This enables users to navigate through various execution iterations (Figure 4B), visually compare outputs, and feed the most suitable results into downstream operators. This method of selecting and visualizing output values streamlines the workflow configuration process by allowing users to evaluate the impact of adjustments made to the operator settings.

The ability to rapidly compare outputs from different iterations directly within the interface also eliminates cumbersome and repetitive code cells that traditionally clutter notebook environments. This could accelerate the decision-making process by allowing users to quickly discern which configurations yield the best results, which could promote a more dynamic and interactive workflow development experience and encourage experimentation and optimization without extensive overhead code duplication.

4 LEARNING, TRANSPARENCY AND COLLABORATION

Terarium integrates each operator with an AI assistant with advanced reasoning-and-action (ReAct) [14] capabilities to interpret user queries and automatically generate executable source code in a relevant programming language. Users

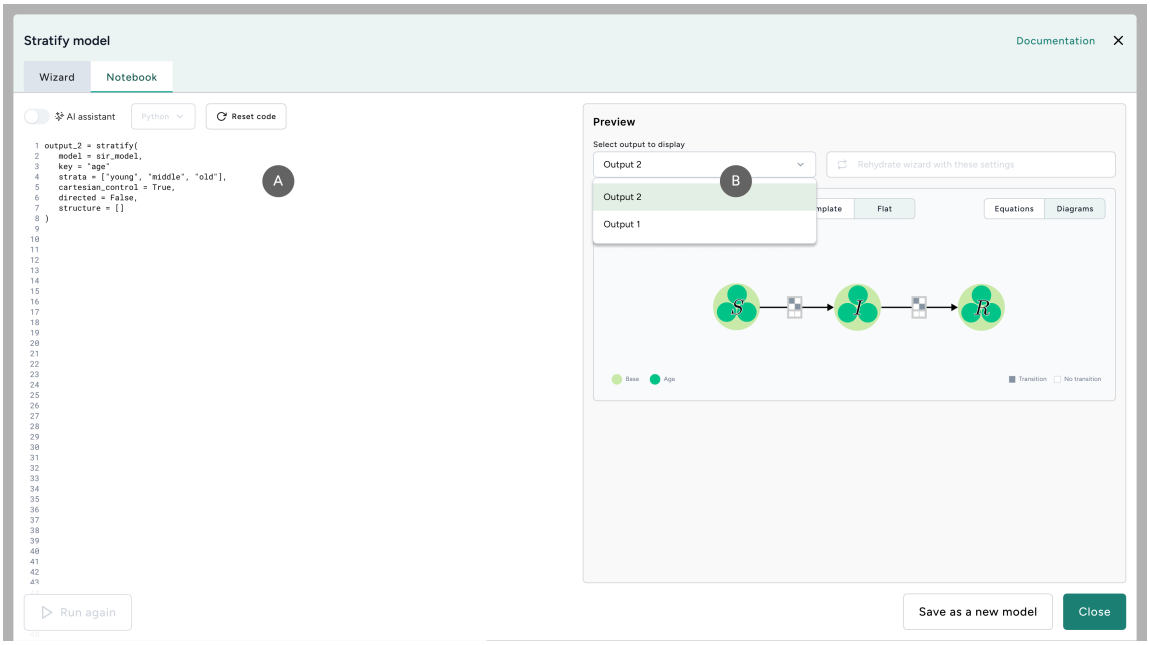

Fig. 4. Notebook view of the 'Stratify' operator. A. The notebook view of Terarium operators aims to give more control over the settings and output of modeling processes by providing a direct coding interface. B. Each output of the operator is saved, enabling users to compare iterations and choose the best results.

are not treated as passive recipients of this automated assistance; they are actively encouraged to inspect, review, edit, and engage in a question-answering conversation with the AI assistant to understand, refine, and extend the code to meet their specific requirements (Figure 5). This is designed to improve the accessibility of intricate computational tasks, making it possible for users with less coding experience to create and optimize sophisticated workflows with fewer steep learning curves.

With code on the left and execution results on the right, the design of Terarium operators prioritizes immediate feedback and code transparency, which are essential for learning [10]. Users can experiment with the code, execute it, see the results of their revisions in real time, and repeat. This immediate responsiveness, coupled with the visibility of code execution and rapid iteration, aims to support a deeper understanding of computational concepts and problem-solving strategies underlying their workflows.

In addition, Terarium promotes collaboration and sharing with the ability to publish and fork projects, facilitating the dissemination of scientific workflows and best practices. Encouraging users to narrate their scientific exploration within workflows could enhance the collaborative experience, aiding others in understanding the rationale behind specific methodologies and contributing improvements. Moreover, each notebook execution in Terarium can be saved and re-applied, allowing scientists to scrutinize both successful and unsuccessful trials, supporting a more comprehensive and transparent understanding of the modeling process. This feature, combined with the other collaborative aspects, should ensure that Terarium notebooks serve not just as computational tools, but also as educational resources that facilitate the sharing of knowledge, reproducibility, and the growth of a collaborative scientific community.

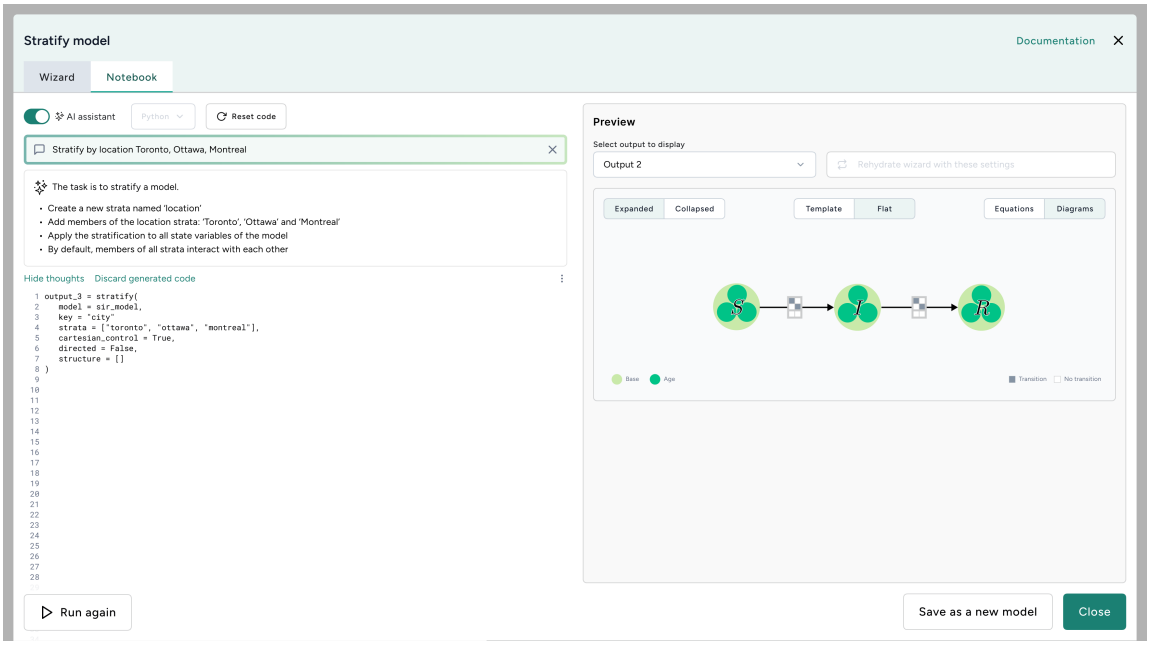

Fig. 5. A 'Stratify' operator with AI-generated code. For users who lack relevant programming experience, an AI assistant provides a method to generate and refine code through natural language prompts. AI reasoning is displayed as human-readable "thoughts" to help users understand how the assistant arrived at its conclusion and whether there are any errors in the output.

5 FUTURE DIRECTIONS

Future research directions include conducting user testing across diverse domains and expertise levels to refine usability and functionality. This is crucial to ensure our approach is effective, user-friendly, and adaptable across various fields and user backgrounds. Additionally, we intend to investigate how we could support 'bring-your-own-operator' in a way that speeds up the integration of new modeling and simulation methodologies, how to integrate AI assistance at the workflow level, and how to enhance narrative capabilities across the notebook environments to potentially elevate the analytical and storytelling power of computational notebooks in Terarium.

6 CONCLUSION

Computational notebooks are widely used tools for writing code within a narrative format. However, they are constrained to short, linear workflows and require significant prior coding experience. Here, we presented Terarium — a platform with the potential of making computational notebooks more scalable, adaptable, user-friendly, and conducive to scientific collaboration. Large scientific modeling workflows and the accompanying problem of kernel state management are tackled by the Workflow Editor, where users can chain together computational notebooks generated with AI assistance as customized operators. The provenance tracking of computation results is supported through the dual-view modality and output selection in each notebook operator, by permitting users to inspect, review, and edit the underlying code. Furthermore, the built-in features around project asset reuse, collaboration and sharing, and learning through code transparency encourage users to leverage prior work from their community to navigate the complexities of such workflows.

ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency (DARPA) [1] under Contract HR001122C0142. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We thank Michael Crouch for helping us review the manuscript.

REFERENCES

[1] Morakot Choetkiertikul, Apirak Hoonlor, Chaiyong Ragkhitwetsagul, Siripen Pongpaichet, Thanwadee Sunetnanta, Tasha Settewong, Vacharavich Jiravatvanich, and Urisayar Kaewpichai. 2023. Mining the Characteristics of Jupyter Notebooks in Data Science Projects. arXiv:2304.05325 [cs.SE]

[2] Hans Fangohr, Thomas Kluyver, and Massimo DiPierro. 2021. Jupyter in Computational Science. *Computing in Science and Engineering* 23, 2 (2021), 5–6. https://doi.org/10.1109/MCSE.2021.3059494

[3] Jesse Harden, Elizabeth Christman, Nurit Kirshenbaum, Mahdi Belcaid, Jason Leigh, and Chris North. 2023. "There is no reason anybody should be using 1D anymore": Design and Evaluation of 2D Jupyter Notebooks. In *Graphics Interface 2023-second deadline.*

[4] Jesse Harden, Elizabeth Christman, Nurit Kirshenbaum, John Wenskovitch, Jason Leigh, and Chris North. 2022. Exploring organization of computational notebook cells in 2d space. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* IEEE, 1–6.

[5] Einblick Analytics Inc. 2023. *Einblick.* Retrieved February 21, 2024 from https://www.einblick.ai/

[6] isee systems Inc. 2024. *STELLA.* Retrieved February 21, 2024 from https://www.iseesystems.com/store/products/stella-architect.aspx

[7] Project Jupyter. 2024. *Jupyter: A Next-Generation Notebook Interface.* Retrieved February 21, 2024 from https://jupyter.org/

[8] Deepthi Raghunandan, Zhe Cui, Kartik Krishnan, Segen Tirfe, Shenzhi Shi, Tejaswi Darshan Shrestha, Leilani Battle, and Niklas Elmqvist. 2024. Lodestar: Supporting rapid prototyping of data science workflows through data-driven analysis recommendations. *Information Visualization* 23, 1 (2024), 21–39. https://doi.org/10.1177/14738716231190429 arXiv:https://doi.org/10.1177/14738716231190429

[9] Bernadette M. Randles, Irene V. Pasquetto, Milena S. Golshan, and Christine L. Borgman. 2017. Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL).* 1–2. https://doi.org/10.1109/JCDL.2017.7991618

[10] Jan Skalka, Martin Drlik, Lubomir Benko, Jozef Kapusta, Juan Carlos Rodríguez del Pino, Eugenia Smyrnova-Trybulska, Anna Stolinska, Peter Svec, and Pavel Turcinek. 2021. Conceptual Framework for Programming Skills Development Based on Microlearning and Automated Source Code Evaluation in Virtual Learning Environment. *Sustainability* 13, 6 (2021). https://doi.org/10.3390/su13063293

[11] Zijie J Wang, Katie Dai, and W Keith Edwards. 2022. Stickyland: Breaking the linear presentation of computational notebooks. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts.* 1–7.

[12] Nathaniel Weinman, Steven M Drucker, Titus Barik, and Robert DeLine. 2021. Fork it: Supporting stateful alternatives in computational notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–12.

[13] Wolfram. 2024. *Wolfram Mathematica.* Retrieved February 21, 2024 from https://www.wolfram.com/mathematica/

[14] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629* (2022).

[1]Distribution Statement A. Approved for public release: distribution is unlimited