Deploying User-Friendly Software: Six Recommendations to Make Single-Cell Foundation Models More Usable For Scientific Discovery

Izumi Ando [1 2] **Hassaan Maan** [* 3 4 5] **Kieran R. Campbell** [* 1 2 4 6 7 8]

Abstract

Foundation models have become ubiquitous in computational biology research. These models take large-scale biological data such as DNA sequences or quantified single-cell RNA molecule counts as input, and learn high-dimensional embeddings that capture patterns in the data through self-supervised training, which are then used for a variety of downstream tasks. Despite their great promise, we found that single-cell foundation models were difficult to install and run, and lacked adequate documentation. To alleviate these problems, we propose six recommendations for better development and deployment of these foundation models to enable ease of use by both computational and non-computational expert end-users, as well as a framework for improving the state of open-source software maintenance in computational biology as a whole.

1. Background

Foundation models are a family of machine learning models that are trained on a vast amount of data, through self-supervised learning (Shwartz-Ziv & LeCun, 2023). From large language models capable of generating text based on prompts, to models trained on specific types of scientific data, foundation models have been developed in various contexts to enhance research in fields including but not limited to material science, biology, and medicine (Pyzer-Knapp et al., 2025; Cui et al., 2025; Khan et al., 2025). In computational biology specifically, foundation models are often trained on data such as DNA and protein sequences (Guo et al., 2025). The rise of single-cell technologies in the 2010s such as single-cell RNA sequencing (scRNA-seq) (Hong & Park, 2020) has also led to a focus in foundation model development in this field as well (Szałata et al., 2024). These models are developed with high hopes of contributing to expediting drug development and biological discovery, as well as creating an all-in-one tool for processing and analyzing high-throughput single-cell data (Szałata et al., 2024; Cui et al., 2024).

However, through analyzing the GitHub repositories and using the implemented software, we found that single-cell foundation models commonly present installation and usability challenges. More specifically, with the speed of innovation often outpacing the discussion and acceptance of ideal development practices, the foundation models had installation and dependency issues, challenges with running and resource management, and were often not maintained after publication. In this paper, we outline the results of our analysis of single-cell foundation model software (Section 2), and discuss practical solutions developers of these models can adopt to address common issues faced by end-users (Section 3). This paper focuses on single-cell foundation models, but we expect the recommendations to be transferable across biological domains.

2. The current landscape of single-cell foundation model software

We investigated single-cell foundation models for a benchmarking study that was orthogonal to this work. Through our attempts to install and run these models, we observed functionality and maintenance issues. We summarize our findings in key ***concerns*** with respect to the foundation model software. These observations serve as the basis of our rationale for the proposed guidelines.

*Jointly supervised this work [1]Lunenfeld-Tanenbaum Research Institute, Toronto, ON, Canada [2]Department of Computer Science, University of Toronto, Toronto, ON, Canada [3]Peter Munk Cardiac Centre, University Health Network, Toronto, ON, Canada [4] Vector Institute, Toronto, ON, Canada [5]Department of Medical Biophysics, University of Toronto, Toronto, ON, Canada [6]Department of Molecular Genetics, University of Toronto, Toronto, Canada [7]Department of Statistical Sciences, University of Toronto, Toronto, ON, Canada [8]Ontario Institute for Cancer Research, Toronto, ON, Canada. Correspondence to: Kieran R. Campbell <kierancampbell@lunenfeld.ca>.

Proceedings of the ICML 2025 Workshop on Championing Open-source Development in Machine Learning (CODEML '25).

2.1. Installation from `requirements.txt` files being insufficient

Many models are designed to be installed via a `requirements.txt` or `setup.py` file. However, this is often not sufficient, as some dependencies cannot be properly installed or have conflicts. This is due to many factors, but a major one is the dynamic nature of dependencies. Python libraries are not static, and as they are updated their usability with other libraries can break. Inputting dependencies, either constrained or unconstrained, in static files with no build tests such as `requirements.txt` or `setup.py`, exposes the model to this failure case.

When these issues arise, the workarounds are time consuming and do not follow a straightforward process. For example, a user could first run a `pip install` using the file in a new virtual environment, check the installed packages, and manually install packages that were left out. Otherwise, the user could manually install all of the dependencies, one by one, with specific version ranges based on trial and error.

2.2. Unresolvable dependency conflicts

Some models contained unresolvable dependency conflicts within their own dependency version ranges. Other models triggered dependency conflicts between the dependencies of the model and the underlying software in the compute system, such as high performance computing (HPC) clusters.

These dependency conflicts are difficult for users to navigate, especially if they are unfamiliar with the libraries needed by the foundation model code. If the dependencies are unresolvable based on user trial and error, this leaves the end-user in a precarious situation, where they have to contact and await response of the code-base authors and maintainers.

2.3. Inactive maintenance of the source code

Many single-cell foundation models are not actively maintained post-publication, in the sense that the main branch of the repository has not been updated for several months or years. We observed GitHub Issues posted on these repositories that have not been addressed. Issues such as dependency conflicts and overall installation problems are prevalent, and if these issues are not addressed, the foundation models are essentially uninstallable for end-users.

2.4. Ambiguous minimum requirements for compute resources

Running single-cell foundation models requires significant computational resources, especially if the models are being fine-tuned, which is required for many of their analyses (Theodoris et al., 2023; Cui et al., 2024). Each foundation model requires a different amount of compute, and the amount of compute also differs based on the tasks that the foundation models are being used for. However, neither the minimum requirements nor a general estimate for the computational cost across tasks were documented for any of the single-cell foundation models we tested.

Experimenting with different resource configurations to determine the optimal setup can be very costly for end-users of these models, especially if they are fine-tuning, which requires a high amount of compute and GPU resources.

2.5. Lack of code documentation

Many of the single-cell foundation models have complex functions and classes with many arguments, but lack comprehensive documentation. Although details of some functions and classes might be outlined in tutorials (for example, in Jupyter notebooks), these will not cover a sufficient amount of the code-base. If end-users need to use functions that are not documented in the tutorials and more comprehensive documentation does not exist, they will be left to their own devices to work through the code. This can be a time consuming and error-prone process.

2.6. Lack of implementation of development and deployment best-practices

We documented our investigation of single-cell foundation models and determined if they follow standard coding best practices, such as package-based deployment, containerization, continuous integration (CI), and documentation. We found that the majority of the single-cell foundation models did not follow these best practices (Table 1), and this analysis serves as the groundwork and rationale for our recommendations which follow. We outline the selection criteria for these models and investigation details in Appendix A.

3. Recommendations for single-cell foundation model software developers

Based on the concerns raised in our analysis of software (Section 2), we have devised six recommendations for single-cell foundation model developers to make these models more usable. We organized the recommendations into three different categories – installation, maintenance, and documentation – corresponding to the process each recommendation aims to improve.

For each recommendation, we provide an overview (high-level description), implementation details (how method developers might go about incorporating these recommendations), and the impact on end-users (why these recommendations would be beneficial to end-users of single-cell foundation models). The latter is especially important, as these recommendations were formulated specifically with end-users of these methods in mind, many of whom might

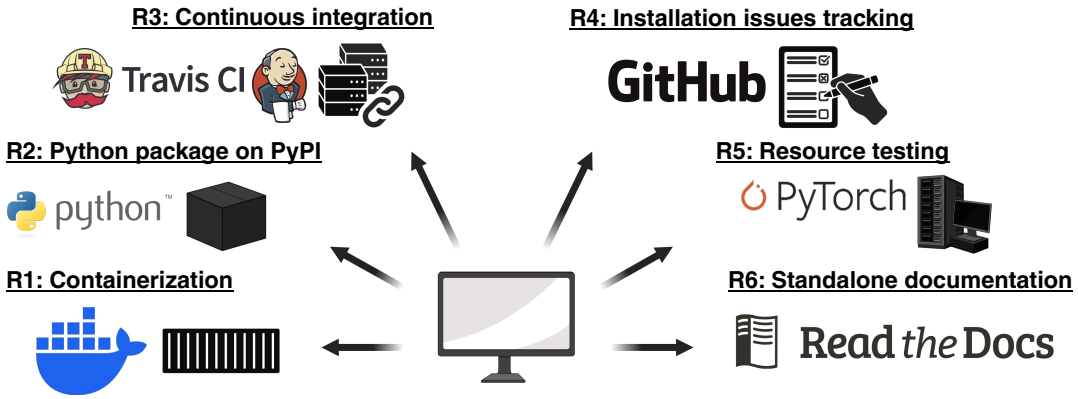

Figure 1. **Overview of recommendations for single-cell foundation model software development.** Each of the recommendations are further outlined, with respective rationale and impact on end-users of the models, in Section 3. The specific tools recommended in the figure are further characterized, with links and documentation, in Appendix B.

Table 1. Implementation of typical best practices in single-cell foundation model software (as of May 2025). **Container**: whether a Docker/Apptainer or similar container image with the foundation model is provided. **Python Pkg**: whether the foundation model and associated code is written as a python package. **CI**: whether continuous integration is practiced based on GitHub Actions or similar tools. **Docs**: whether standalone documentation is available for the model.

Model	Container	Python Pkg	CI	Docs
scBERT	×	×	×	×
Geneformer	×	×	×	√
UCE	×	×	√	×
scGPT	×	√	×	√
scFoundation	×	×	×	×
scLong	×	×	×	×
CellPLM	×	√	×	×
SCimilarity	√	√	×	√
tGPT	×	×	×	×
CellLM	√	×	×	×
GeneCompass	×	×	×	×
GenePT	×	×	×	×
Total	**2/12**	**3/12**	**1/12**	**3/12**

not have strong computational expertise to navigate the issues outlined in Section 2. **Links to specific recommended software and tools can be found in Appendix B.**

3.1. Foundation model installation

3.1.1. Recommendation 1: Deploying the model in a container

Overview: Containerizing the foundation model environment is potentially a one-size-fits-all solution for the machine-dependent dependency conflicts that arise when users attempt to install foundation models within the limitations of conda or virtual environments, because properly containerized software will run regardless of the underlying dependencies and environments in the user's machine.

Implementation: Providing Docker and Apptainer container images as an alternative installation method should be standard practice when deploying single-cell foundation models, as well as any foundation model expected to be used in a context with multiple programming languages.

Impact on end-users: Fully containerized models that do not require installation of several packages to behave appropriately with underlying dependencies will allow for easier and more reproducible installation across systems. This is especially true for HPC systems, which often have a high variation in their underlying operating systems and libraries, and are a typical choice for scientific computing.

3.1.2. Recommendation 2: Deploying the model as a python package

Overview: Many single-cell foundation models are deployed with a `requirements.txt` or `setup.py` file for installation, which are suboptimal as outlined in Section 2. Thus, deploying foundation models as a Python package on PyPI would be a beneficial design choice.

Implementation: Deploying foundation models and their associated code for tasks such as fine-tuning, as a Python package on PyPI, and allowing installation via pip. Creation of python packages can be done using several frameworks, including the Poetry python package manager.

Impact on end-users: Managing dependencies is easier when all dependencies of the foundation model can be handled with a single pip call from PyPI. Furthermore, version control and resolving dependency conflicts is much easier with a validated python package compared to a repository.

3.2. Foundation model maintenance

3.2.1. Recommendation 3: Continuous integration of foundation model software

Overview: Frequently used software will often receive feedback from users requesting new features or fixes for issues that arise. These frequent changes can cause software to break. To make sure only safe changes are being deployed, developers should practice continuous integration (CI).

Implementation: By using CI systems such as GitHub Actions or Travis CI, every time an update is made to the main software repository, tests to make sure the software is building as a whole and functioning as intended are triggered automatically, and the results of these tests are publicly displayed on the repository to indicate to both developers and users that the software is installable. Additionally, with GitHub Actions, the build tests can be configured to be run on different operating systems to make sure they are usable across different HPC systems.

Impact on end-users: Single-cell foundation model software that has CI leads to better guarantees for end-users installing them on diverse platforms. Further, common workflows in single-cell such as cell-type annotation can be made more robust by having tests associated with them that need to be passed through CI. This leads to higher confidence from end-users of the models that the associated code is error-free and will generalize to novel datasets.

3.2.2. Recommendation 4: Set up a forum for dependency and installation issues

Overview: Installing foundation models that are neither containerized nor wrapped as a Python package can be a challenging task. In a case where neither of the two above solutions can be provided, a low-effort solution could be to provide a forum-like platform to list dependency version combinations that have worked or failed for various users.

Implementation: The most common forum associated with code challenges is GitHub Issues. However, having several distinct issues for dependency challenges is not an optimal setup. Techniques to track dependency issues, based on packages, operating systems, and underlying libraries, should be utilized. The most simple version of this would be a pinned GitHub Issues thread that tracks current conflicts and aims to address them based on distinct pull requests and interaction of end-users and developers.

Impact on end-users: This setup is especially helpful when certain version ranges of packages are not available in their compute environment and the user must find alternative versions that are compatible with the rest of the dependencies. In general, if dependency issues and system conflicts are tracked in a methodical and centralized manner, this will lead to quicker solutions and allow end-users to determine if their systems may be currently incompatible and what dependencies are causing issues with their installations.

3.3. Foundation model documentation

3.3.1. Recommendation 5: Explicitly listing compute resource requirements

Overview: Adding explicit compute, memory, and GPU requirements for zero-shot inference, fine-tuning and downstream analyses of single-cell foundation model methods.

Implementation: Developers should test the minimum requirements of compute resources such as memory, number of GPUs, number of CPUs allocated per task, and time allowance needed to successfully run the foundation model, and clearly indicate these numbers in the documentation along with guidelines for calculating any adjustments the user would need to make depending on their use case.

Impact on end-users: The amount of resources required to run foundation models are not obvious, yet rarely documented. Many researchers use public HPC systems, or work with a limited allowance of a private system such as Google Cloud Platform, with restrictions on resource usage. Thus, when attempting to run the foundation models with different resource requirements, they are at risk of wasting time, money, and energy. Explicitly stating the compute requirements is a preventive measure for resource waste.

3.3.2. Recommendation 6: Include comprehensive standalone documentation

Overview: Detailed documentation with instructions on how to use the software as well as sample code with sample datasets should always be included with high code base coverage. This documentation should be standalone, and not simply part of the repository README, as they are not designed for tutorials and code APIs.

Implementation: We recommend using *Read the Docs*, which is a platform that can host documentation that integrates with GitHub. It provides a framework for developers to organize their documentation, as well as built-in functionality such as search and navigation.

Impact on end-users: Comprehensive standalone documentation allows end-users to search APIs to resolve their issues and questions in a succinct and self-contained manner. This avoids scattered documentation across multiple READMEs and notebook pages in a repository, which is often difficult to navigate and may lead to attrition in the processes of installing and running the single-cell foundation models.

4. Discussion

Due to competition within respective academic fields, there is great incentive to publish with speed. As a consequence, software may not be developed with high standards for quality, reproducibility, and usability. This phenomenon, along with its repercussions of reduced reproducibility are widely acknowledged (Leprevost et al., 2014; Ziemann et al., 2023). Identifying specific issues and devising practical recommendations as we have done in this work is only the first step to making tangible improvements. Future directions of our work include but are not limited to: 1) Collecting feedback on these recommendations from foundation model users and developers, 2) Creating a website to guide new developers who are trying to implement the recommendations and to showcase successful implementation cases, 3) Automating the implementation of the six recommendations using large language models, and 4) Raising awareness and encouraging more academic developers to implement these recommendations. Furthermore, in Appendices C and D, we outline work related to our recommendations, existing solutions to the concerns we raise, and discuss our own proposal for a more sustainable computational biology research ecosystem.

Impact Statement

This paper presents work whose goal is to advance the fields of Machine Learning and Computation Biology. Development of single-cell foundation models is a major and growing research direction in the field of Computational Biology. However, we highlight that the software for these models is often not in a state that best enables usability and uptake, especially by non-computational experts, which comprise a significant amount of end-users, such as those from biology and medicine backgrounds. These issues can also hamper reproducibility of analyses with these models.

As foundation models often need a very high amount of compute and data resources to train and validate, it is imperative that the Computational Biology and Machine Learning communities work towards more user-friendly software, that will enable better benchmarking of these models to determine their true efficacy, and usage by experts from different domains to help solve their pressing challenges. From a societal perspective, development of single-cell and other biological foundation models adherent to the guidelines outlined in the paper will advance research in key disease settings with significant human burden (e.g. cancer, cardiovascular disease, neuro-degenerative diseases), in a more reproducible and robust manner, than compared to software that is poorly documented and difficult to install and run.

References

Artaza, H., Chue Hong, N., Corpas, M., Corpuz, A., Hooft, R., Jimenez, R. C., Leskošek, B., Olivier, B. G., Stourac, J., Svobodová Vařeková, R., Van Parys, T., and Vaughan, D. Top 10 metrics for life science software good practices. *F1000Res.*, 5:2000, August 2016.

Bai, D., Mo, S., Zhang, R., Luo, Y., Gao, J., Yang, J. P., Wu, Q., Singh, D., Rahmani, H., Amariuta, T., Grotjahn, D., Zhong, S., Lewis, N., Wang, W., Ideker, T., Xie, P., and Xing, E. ScLong: A billion-parameter foundation model for capturing long-range gene context in single-cell transcriptomics. *bioRxiv*, November 2024.

Barker, M., Chue Hong, N. P., Katz, D. S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., and Honeyman, T. Introducing the FAIR principles for research software. *Sci. Data*, 9(1):622, October 2022.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel,

K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A., Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models. *arXiv [cs.LG]*, August 2021.

Chan Zuckerberg Initiative. CZI – essential open source software for science, a. URL https://chanzuckerberg.com/eoss/.

Chan Zuckerberg Initiative. Early access to CZI's AI-powered cell models, b. URL https://chanzuckerberg.com/blog/ai-cell-models-platform-release/.

Chen, Y. and Zou, J. Simple and effective embedding model for single-cell biology built from ChatGPT. *Nat. Biomed. Eng.*, 9(4):483–493, April 2025.

Clyburne-Sherin, A., Fei, X., and Green, S. A. Computational reproducibility via containers in social psychology. *Meta-Psychology*, 3, 2019.

Cui, H., Wang, C., Maan, H., Pang, K., Luo, F., Duan, N., and Wang, B. scGPT: toward building a foundation model for single-cell multi-omics using generative AI. 21 (8):1470–1480, 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02201-0. URL https://www.nature.com/articles/s41592-024-02201-0. Publisher: Nature Publishing Group.

Cui, H., Tejada-Lapuerta, A., Brbić, M., Saez-Rodriguez, J., Cristea, S., Goodarzi, H., Lotfollahi, M., Theis, F. J., and Wang, B. Towards multimodal foundation models in molecular cell biology. 640(8059): 623–633, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-08710-y. URL https://www.nature.com/articles/s41586-025-08710-y. Publisher: Nature Publishing Group.

Eustace, S. and contributors, T. P. Poetry: Python packaging and dependency management made easy, 2025. URL https://python-poetry.org. If you use this software, please cite it using the metadata from this file.

Guo, F., Guan, R., Li, Y., Liu, Q., Wang, X., Yang, C., and Wang, J. Foundation models in bioinformatics. 12 (4):nwaf028, 2025. ISSN 2095-5138. doi: 10.1093/nsr/nwaf028. URL https://doi.org/10.1093/nsr/nwaf028.

Hao, M., Gong, J., Zeng, X., Liu, C., Guo, Y., Cheng, X., Wang, T., Ma, J., Zhang, X., and Song, L. Large-scale foundation model on single-cell transcriptomics. *Nat. Methods*, 21(8):1481–1491, August 2024.

Heimberg, G., Kuo, T., DePianto, D. J., Salem, O., Heigl, T., Diamant, N., Scalia, G., Biancalani, T., Turley, S. J., Rock, J. R., Corrada Bravo, H., Kaminker, J., Vander Heiden, J. A., and Regev, A. A cell atlas foundation model for scalable search of similar human cells. *Nature*, 638 (8052):1085–1094, February 2025.

Helical Team. helicalai/helical: v1.1.0, November 2024. URL https://doi.org/10.5281/zenodo.13135902.

Hong, T. H. and Park, W.-Y. Single-cell genomics technology: perspectives. 52(9):1407–1408, 2020. ISSN 2092-6413. doi: 10.1038/s12276-020-00495-6. URL https://www.nature.com/articles/s12276-020-00495-6. Publisher: Nature Publishing Group.

Jiménez, R. C., Kuzak, M., Alhamdoosh, M., Barker, M., Batut, B., Borg, M., Capella-Gutierrez, S., Chue Hong, N., Cook, M., Corpas, M., Flannery, M., Garcia, L., Gelpí, J. L., Gladman, S., Goble, C., González Ferreiro, M., Gonzalez-Beltran, A., Griffin, P. C., Grüning, B., Hagberg, J., Holub, P., Hooft, R., Ison, J., Katz, D. S., Leskošek, B., López Gómez, F., Oliveira, L. J., Mellor, D., Mosbergen, R., Mulder, N., Perez-Riverol, Y., Pergl, R., Pichler, H., Pope, B., Sanz, F., Schneider, M. V., Stodden, V., Suchecki, R., Svobodová Vařeková, R., Talvik, H.-A., Todorov, I., Treloar, A., Tyagi, S., van Gompel, M., Vaughan, D., Via, A., Wang, X., Watson-Haigh, N. S., and Crouch, S. Four simple recommendations to encourage best practices in research software. *F1000Res.*, 6: 876, June 2017.

Khan, W., Leem, S., See, K. B., Wong, J. K., Zhang, S., and Fang, R. A comprehensive survey of foundation models in medicine, 2025. URL http://arxiv.org/abs/2406.10729.

Leprevost, F. d. V., Barbosa, V. C., Francisco, E. L., Perez-Riverol, Y., and Carvalho, P. C. On best practices in the development of bioinformatics software. 5, 2014.

ISSN 1664-8021. doi: 10.3389/fgene.2014.00199. URL https://www.frontiersin.orghttps://www.frontiersin.org/journals/genetics/articles/10.3389/fgene.2014.00199/full. Publisher: Frontiers.

Liu, T., Li, K., Wang, Y., Li, H., and Zhao, H. Evaluating the utilities of foundation models in single-cell data analysis. *bioRxivorg*, December 2024.

Pyzer-Knapp, E. O., Manica, M., Staar, P., Morin, L., Ruch, P., Laino, T., Smith, J. R., and Curioni, A. Foundation models for materials discovery – current state and future directions. 11(1): 1–10, 2025. ISSN 2057-3960. doi: 10.1038/s41524-025-01538-0. URL https://www.nature.com/articles/s41524-025-01538-0. Publisher: Nature Publishing Group.

Rosen, Y., Roohani, Y., Agrawal, A., Samotorcan, L., Consortium, T. S., Quake, S. R., and Leskovec, J. Universal cell embeddings: A foundation model for cell biology. pp. 2023–11, 2023. URL https://www.biorxiv.org/content/10.1101/2023.11.28.568918.abstract. Publisher: Cold Spring Harbor Laboratory.

Shen, H., Liu, J., Hu, J., Shen, X., Zhang, C., Wu, D., Feng, M., Yang, M., Li, Y., Yang, Y., Wang, W., Zhang, Q., Yang, J., Chen, K., and Li, X. Generative pretraining from large-scale transcriptomes for single-cell deciphering. *iScience*, 26(5):106536, May 2023.

Shwartz-Ziv, R. and LeCun, Y. To compress or not to compress- self-supervised learning and information theory: A review, 2023. URL http://arxiv.org/abs/2304.09355.

Szałata, A., Hrovatin, K., Becker, S., Tejada-Lapuerta, A., Cui, H., Wang, B., and Theis, F. J. Transformers in single-cell omics: a review and new perspectives. 21(8):1430–1443, 2024. ISSN 1548-7105. doi: 10.1038/s41592-024-02353-z. URL https://www.nature.com/articles/s41592-024-02353-z. Publisher: Nature Publishing Group.

Theodoris, C. V., Xiao, L., Chopra, A., Chaffin, M. D., Al Sayed, Z. R., Hill, M. C., Mantineo, H., Brydon, E. M., Zeng, Z., Liu, X. S., and Ellinor, P. T. Transfer learning enables predictions in network biology. 618(7965):616–624, 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06139-9. URL https://www.nature.com/articles/s41586-023-06139-9. Publisher: Nature Publishing Group.

Wen, H., Tang, W., Dai, X., Ding, J., Jin, W., Xie, Y., and Tang, J. CellPLM: Pre-training of cell language model beyond single cells. *bioRxiv*, October 2023.

Yang, F., Wang, W., Wang, F., Fang, Y., Tang, D., Huang, J., Lu, H., and Yao, J. scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. *Nat. Mach. Intell.*, 4(10):852–866, September 2022.

Yang, X., Liu, G., Feng, G., Bu, D., Wang, P., Jiang, J., Chen, S., Yang, Q., Miao, H., Zhang, Y., Man, Z., Liang, Z., Wang, Z., Li, Y., Li, Z., Liu, Y., Tian, Y., Liu, W., Li, C., Li, A., Dong, J., Hu, Z., Fang, C., Cui, L., Deng, Z., Jiang, H., Cui, W., Zhang, J., Yang, Z., Li, H., He, X., Zhong, L., Zhou, J., Wang, Z., Long, Q., Xu, P., X-Compass Consortium, Wang, H., Meng, Z., Wang, X., Wang, Y., Wang, Y., Zhang, S., Guo, J., Zhao, Y., Zhou, Y., Li, F., Liu, J., Chen, Y., Yang, G., and Li, X. GeneCompass: deciphering universal gene regulatory mechanisms with a knowledge-informed cross-species foundation model. *Cell Res.*, 34(12):830–845, December 2024.

Zhao, S., Zhang, J., and Nie, Z. Large-scale cell representation learning via divide-and-conquer contrastive learning. *arXiv [cs.CE]*, June 2023.

Ziemann, M., Poulain, P., and Bora, A. The five pillars of computational reproducibility: bioinformatics and beyond. 24(6):bbad375, 2023. ISSN 1467-5463. doi: 10.1093/bib/bbad375. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10591307/.

A. Selection criteria for single-cell foundation models and survey details

The term ”foundation model” does not have a strict and well-agreed upon definition in the computational biology community, but for the selection of the models in our analysis, we use the definition from Bommasani et al. (2021):

> *A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks...*

Given this criteria, we selected the following models that meet the following two requirements:

1. Training on a large amount of data in a self-supervised manner, or using models that have trained on a large corpus of data in a similar manner (e.g. ChatGPT)
2. Aim to learn representations that are useful for a variety of downstream tasks in single-cell analysis

This led us to include the following models into our analysis:

- scBERT (Yang et al., 2022)
- Geneformer (Theodoris et al., 2023)
- Universal Cell Embeddings (UCE) (Rosen et al., 2023)
- scGPT (Cui et al., 2024)
- scFoundation (Hao et al., 2024)
- scLong (Bai et al., 2024)
- CellPLM (Wen et al., 2023)
- SCimilarity (Heimberg et al., 2025)
- tGPT (Shen et al., 2023)
- CellLM (Zhao et al., 2023)
- GeneCompass (Yang et al., 2024)
- GenePT (Chen & Zou, 2025)

While we did our best to accommodate models that fit the outlined criteria, we understand that this is not an exhaustive list, and that the landscape of single-cell foundation models is constantly evolving. For searching, we used a combination of the models outlined in Liu et al. (2024), and search terms encompassing the previously outlined criteria for the definition of single-cell foundation models.

For the criteria indicated in Table 1, we searched the model code and repositories in the following manner:

1. **Container**: For containerization, we located the GitHub repository of each model, and went through the README and any associated links (such as further documentation) to determine if Docker, Apptainer, or similar containerization solutions were present.
2. **Python Pkg**: Similar to containerization, we determined if a python package was released by going through the GitHub repository of each model, including the READMEs.
3. **CI**: For continuous integration, we determined if any badges relevant to continuous integration (e.g. Travis CI) were present in the code repository and/or if GitHub integrated CI such as precommit or GitHub Actions were present in the repository.
4. **Docs**: To determine if standalone documentation was present, we went through the GitHub repositories of the models, including the READMEs, and other subsections such as the ‘About’ portion, where documentation links are typically found.

B. Details and links for tools outlined in recommendations

Trademark notice: All product names, logos, and brands shown in Figure 1 are trademarks or registered trademarks of their respective owners and are used here for identification purposes only. Use of these names, logos, and brands does not imply endorsement. All rights reserved.

We compile a list of resources we outlined for development best-practices, as well as a short description of their usage, sorted based on the specific recommendation. We include additional resources here for a more comprehensive list.

B.1. Recommendation 1

Docker: Docker is an all-in-one tool for containerizing software, applications, and associated workflows. https://docs.docker.com/

Apptainer: Formerly known as Singularity, is an alternative containerization tool that is often better compatible with HPC systems. https://apptainer.org/

B.2. Recommendation 2

Python packages guide: A dedicated guide from the Python Software Foundation on creating and deploying python packages. https://packaging.python.org/en/latest/tutorials/packaging-projects/

PyPI: The Python Package Index - a repository for developers to upload python packages such that they can be easily installed via Pip. https://pypi.org/

Poetry: An all-in-one method for managing dependencies and creating and publishing python packages (Eustace & contributors, 2025), with cross operating system compatibility, including Windows, Linux, and macOS. https://python-poetry.org/

B.3. Recommendation 3

Github Actions: A GitHub compatible tool for continuous integration and development. Can be used for build tests across platforms (e.g. different python versions) and operating systems, automatic testing on pushes to certain branches, and more. Free up to a usage limit. https://github.com/features/actions

Travis CI: A continuous integration tool that aims to simplify syntax, and has several powerful integrations. https://www.travis-ci.com/product/

Jenkins: An open-source and free platform for continuous integration and development, but requires self-hosting. https://www.jenkins.io/

B.4. Recommendation 4

GitHub Issues: A tool integrated within GitHub to track and manage Issues, including dependency and installation problems. Issues can be sorted based on importance and type, and can be interlinked. https://docs.github.com/en/issues

GitLab Issues: Similar to GitHub Issues, but focused on developers that use GitLab and integrates easily with many associated tools. https://docs.gitlab.com/user/project/issues/

Jira: Enterprise-level tool for project management, which includes tracking software issues and bugs. https://www.atlassian.com/software/jira

B.5. Recommendation 5

NVIDIA-smi: Tool from NVIDIA to monitor GPU usage, GPU memory allocation, power usage, and other GPU-related factors. https://docs.nvidia.com/deploy/nvidia-smi/index.html

Torch Profiler: A built-in PyTorch tool/API that allows for tracing of code execution and determining memory usage and time allocation. https://docs.pytorch.org/docs/stable/profiler.html

psutil library: A comprehensive library for tracking CPU usage, memory usage, disk input/output, and more. Generally

applicable to monitoring many different types of processes. https://github.com/giampaolo/psutil

Weights and Biases (Wandb): A toolkit and framework for monitoring deep-learning experiments and training runs, that is also able to monitor system memory and GPU utilization. https://wandb.ai/wandb_fc/articles/reports/Monitor-Improve-GPU-Usage-for-Model-Training--Vmlldzo1NDQzNjM3

B.6. Recommendation 6

Read the Docs: A tool for hosting code API and tutorial documentation that is generated by methods such as Mkdocs (https://www.mkdocs.org/) and Sphinx (https://www.sphinx-doc.org/en/master/). Supports several important features, such as search, GitHub integration, automatic documentation building and updating, and more. https://docs.readthedocs.com/platform/stable/index.html

GitHub Pages: An alternative to Read the Docs for hosting documentation, but lacks key features such as search which would have to be added in manually. https://pages.github.com/

C. Current solutions and related work addressing software issues in computational biology

The following Appendix section summarizes related work in terms of publications and guidelines that aim to address challenges in computational scientific software development (Table 2), specifically focused on computational biology, as well as services that aim to aide researchers in developing, deploying, and using computational tools (Table 3).

Table 2. Summary of Related Work and Recommendations for Scientific Software Development and Deployment

Work	Category	Details	Strengths	Limitations
5 Pillars of Computational Reproducibility (Ziemann et al., 2023)	Paper	An article that provides a framework to make computational research more reproducible.	General, adaptable to many fields and generations of technology.	Not focused on foundation model research, which presents several of its own distinct challenges.
FAIR Biomedical Research Software (FAIR-BioRS) Guidelines https://github.com/FAIR-BioRS/Docs	Guidelines	Biomedical research software-focused guidelines for improving accessibility, find-ability, interoperability, and re-usability.	Based on FAIR4RS software principles (Barker et al., 2022), which are a grounded set of general principles for research software.	Not specific to single-cell foundation models, and since the guidelines are older, are not specific to the challenges of foundation model software in general.
ELIXIR Group Four simple recommendations to encourage best practices in research software (Jiménez et al., 2017)	Guidelines	Four general recommendations to make research software more reusable, transparent, and discoverable.	Strong set of generally applicable guidelines for developers.	Lacks detail on key aspects of reproducibility and re-usability, such as continuous integration, and is overall very general and not at all specific to foundation model software.
ELIXIR Group Top 10 metrics for life science software good practices (Artaza et al., 2016)	Guidelines	10 key recommendations for life science based software, which include version control, automated testing, and documentation.	A strong set of metrics and indicators of good software, which extend quite far into the development (not just deployment) realm, including code reviews and licensing. Quite a few recommendations overlap with our work, which is sensible given that good software practice does not necessarily need novel frameworks	Not specific to machine learning and deep learning libraries, which present unique challenges and need additional recommendations on factors such as resource usage and the need for containerization when foundation models are integrated into larger single-cell and bioinformatics workflows.

Table 3. Summary of Solutions to Improve Scientific Software Reproducibility and Accessibility

Solution	Category	Details	Strengths	Limitations
Essential Open Source Software for Science	Grants	Grant by the Chan Zuckerberg Initiative to fund initiatives to make open source software more sustainable (Chan Zuckerberg Initiative, a).	Flexible solution to make better open-source practices more sustainable.	—
Helical	Services	Python package that wraps biological foundation models in a uniform interface (Helical Team, 2024).	Easy to use and contains representative models in the field.	Restricted to foundation models at the moment.
Code Ocean	Services	Platform that hosts a wide range of bioinformatics software in a ready to use state (Clyburne-Sherin et al., 2019).	Makes software accessible to non-computational scientists.	Some user tiers are not free.
Garden	Services	A platform which hosts AI models in a containerized environment to allow researchers to run them within a limited GPU allowance without having to install and set up in their own machine. Also provides guidance for developers to publish their model on the platform `https://thegardens.ai/`.	Centralized support for many models, provides a good framework for model development in general, makes models more accessible in general.	Restricted to AI models.
Platform for Cell Models	Services	A platform currently under development by the Chan Zuckerberg Initiative that will contain documentation, datasets, and Google Colab tutorials to provide easy access to curated AI models (Chan Zuckerberg Initiative, b).	Makes models more accessible in general.	Restricted to AI models.

D. Establishing a centralized software maintenance organization

The solutions listed in Appendix C are impactful and effective in their own niches. However, to create a solution that takes a more holistic viewpoint to improve the state of code maintenance in computational biology, the following three aspects should be addressed:

1. Acknowledge and work around the established norms and systems of academic research.
2. Set incentives for those contributing to the initiative.
3. Promote the initiative with the goal of making it a go-to solution.

A potential solution that has yet to be explored is creating an organization that solely focuses on taking over the "maintenance" role of academic software. The original developers of the software would onboard the staff members to help them navigate the code and to help with critical decision making, but would retain the credit for the software (Front #1). Funding can be obtained through grants such as the Essential Open Source Software for Science by the Chan Zuckerberg Initiative (Chan Zuckerberg Initiative, a), sponsorship from corporations, or even through an affordable subscription model targeting research institutions (Front #2). Finally, by developing a strong presence in the field by having representatives attend conferences, investing in high quality digital media output, and getting institutional leaders on board with the organization's work, the initiative could become a centralized solution to this issue (Front #3). Figure 2 visualizes the relationships between the stakeholders in this idea.

Determining the viability of this idea would require further discussion and research regarding the stakeholders in the system as well as the amount of resources required to set up and maintain such a system. However, it is also important to propose these ideas as a starting point for discussion in the community, given the observed issues with respect to software maintenance outlined in this work and others.

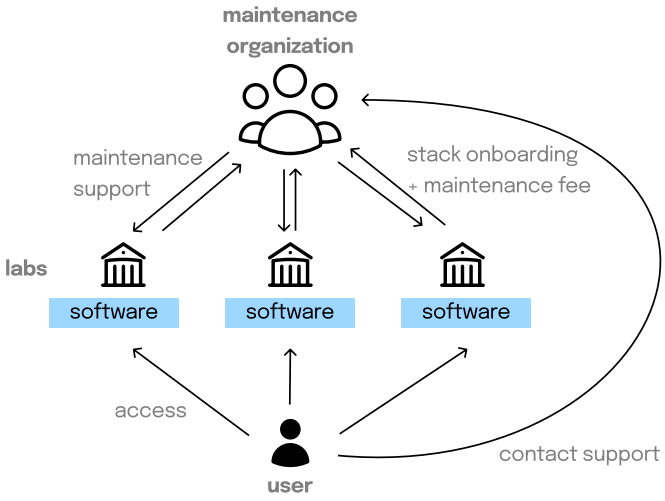

Figure 2. Rudimentary model of proposed centralized software maintenance system.