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

Link to Paper

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

*Jointly supervised this work, **1:** Lunenfeld-Tanenbaum Research Institute at Sinai Health, **2:** Department of Computer Science at the University of Toronto, **3:** Peter Munk Cardiac Centre at University Health Network, **4:** Vector Institute, **5:** Department of Medical Biophysics at the University of Toronto, **6:** Department of Molecular Genetics at the University of Toronto, **7:** Department of Statistical Sciences at the University of Toronto, **8:** Ontario Institute for Cancer Research

TL;DR

- **Problem:**
  - Most single-cell foundation models are not developed using industry standard software development practices.
  - Many are difficult to use, sometimes even not installable.
- **Why we should care:**
  - Users cannot use these foundation models to further scientific discovery.
  - Leads to wasted time, computational resources, and work.
- **What we (developers) should do:**
  - Aim to implement the 6 recommendations below when deploying foundation models.

Common Pitfalls

❌ insufficient `requirements.txt` installation set up

❌ unresolvable dependency conflicts

❌ inactive maintenance

❌ ambiguous compute resource requirements

❌ lack of documentation

Table 1: **Implementation of typical best practices in single-cell foundation model software (as of May 2025).** This data serves as our rationale for our recommendations.

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**

Impact / Further Implementation Suggestions

1. Removes need for users to resolve dependency conflicts.
   - Useful for models used in context with multiple programming languages.
2. Improves user experience of installing foundation models.
3. Make sure only safe changes are being deployed as the software gets updated over time.
4. A low effort solution for models that are neither containerized nor wrapped as a Python package.
5. List minimum requirements for memory, number of GPUs, number of CPUs allocated per task, and time allowance needed to successfully run the foundation model.
6. Documentation should have high code coverage.

Efforts Addressing Software Issues in Computational Biology

Current Existing Solutions

Grants

Essential Open Source Software for Science
Grant by the Chan Zuckerberg Initiative to fund initiatives to make open source software more sustainable.

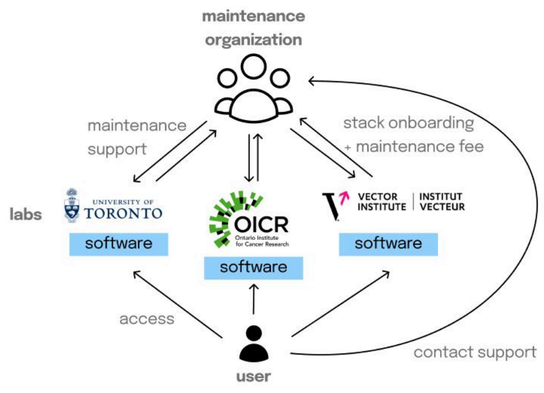
Services

Helical AI
Python package that wraps biological foundation models in a uniform interface.

Garden
A platform that hosts AI models in a containerized environment to allow researchers to run them within a limited GPU allowance.

Code Ocean
Platform that hosts a wide range of bioinformatics software in a ready to use state.

Proposal: Centralized Software Maintenance



An organization solely dedicated to the maintenance of important software could could be a sustainable solution to keep academic software usable without disrupting the established norms and incentives in academia. Potential funding sources include grants and affordable fees collected from journals/labs/users.

Future Directions

- Automating the implementation of the 6 recommendations using LLMs.
- Collecting feedback from foundation model users and developers on the recommendations.
- Creating a website to guide new developers trying to implement the recommendations, and to showcase successful implementations.

References

Szałata, A., Hrovatin, K., Becker, S., Tejada-Lapuerta, A., Cui, H., Wang, B., & Theis, F. J. (2024). Transformers in single-cell omics: A review and new perspectives. Nature Methods, 21(8), 1430–1443. https://doi.org/10.1038/s41592-024-02353-z

Ziemann, M., Poulain, P., & Bora, A. (2023). The five pillars of computational reproducibility: Bioinformatics and beyond. Briefings in Bioinformatics, 24(6), bbad375. https://doi.org/10.1093/bib/bbad375