Standardization in computational topology

Jānis Lazovskis, University of Latvia

Open problem session
ICERM, May 20, 2026

Standardization in computational topology

Personal context: An issue I have struggled with for the past couple years, especially this year

---

General context: You find / create a mathematical method, and you want to make a computer do it (for exploration, publication, collaboration, etc). Then...

one of the following happens (in no particular order):

- You implement it yourself without issues
- You get someone else to do it, they implement it
- You attempt to implement it yourself, it takes longer than expected
- You get someone to help you implement it, it takes longer than expected
- You are unable to implement it exactly, you find an existing, not-quite-the-same method
- You are unable to implement it exactly, you give up

Reasons: Obscure code, incompatible packages, unclear / misleading documentation, ...

Standardization in computational topology

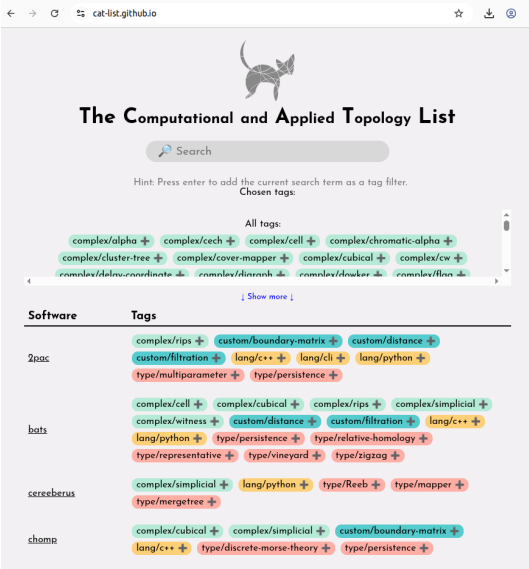
Technical reasons for not succeeding: Code you work with often requires specific data types, assumes some PH steps have been done, or executes the PH pipeline in its own particular way:

- reading in / transforming the data
- reducing the matrix
- getting persistence pairs / representative cycles

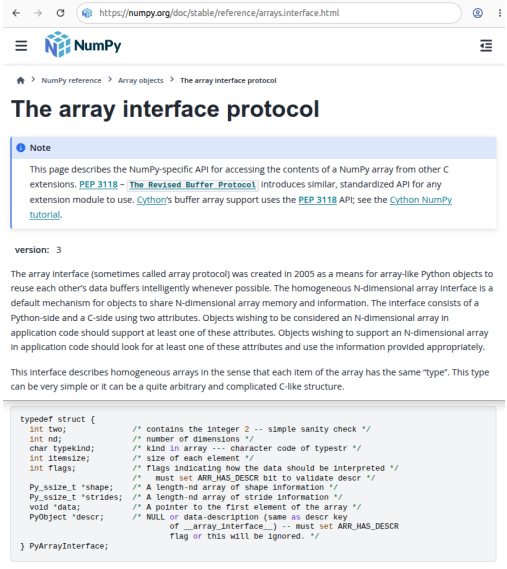
This "common stuff" makes it harder for you to implement your "interesting stuff"

Problem: Every PH software does the basic (PH) operations from scratch, in its own way.
Users unfamiliar with the software waste time* in adjusting, redoing, verifying.
Barriers grow to getting new people involved.

Would be great if: You could focus on implementing your specific contribution, without making annoying wrapper / helper functions, knowing the precise data types of what you need



The current situation



An analogous solution

Defining a protocol

Proposition: Our community should have a **collection of rules** that describes the types of "common data" used in computational topology, and specifies their properties.

- *Similar to NumPy's dense matrix interface protocol, but for sparse matrices*
- *Similar* to OAT's oracles for user queries, but at the level of computer queries*

Simple (personally suggested) example:

- A *matrix* is a *dictionary* with $\mathbf{Z}_{\geqslant 0}$ indices and non-empty $\mathbf{Z}_{\geqslant 0}$-*vectors*.
- A *barcode* is an *unordered set* whose elements are *pairs* of elements of $\mathbf{Z}_{\geqslant 0}$.

Things to keep in mind:

- People who implement major packages should have an open discussion about the different technical issues. *Who are these people? How can they be brought together?*
- Lots of software is already developed, people are happy with their own setup. *How can people be convinced this is worthwhile?*