The Journal of Open Source Software

Episimlab: A Python package for modeling epidemics

Ethan Ho[1] **and Kelly Pierce**[1]

1 Texas Advanced Computing Center (TACC) - University of Texas at Austin

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

- Epidemic modeling is important
  - COVID (review ref)
  - But also cybersecurity (review ref)
- `Episimlab` is a Python package for development and execution of complex compartmental epidemic models.
- Users can
  - Quickly get up and running with a model off the shelf
  - Add complexity to the model by adding or modifying modular components, known as `processes` in Episimlab
  - It provides a framework that enables users to develop models that are modular, extensible, reusable, and reproducible.
- Episimlab ships with performace-optimized implementations of commonly used disease modeling routines, such as the force of infection (FOI) calculation.

Statement of need

- Developing epidemic models is time consuming and rarely compostable/reproducible (review ref?)
  - Subject matter experts such as epidemiologists often recapitulate routines that are common in compartmental epidemic models, such as calculating the force of infection.

`Episimlab` was designed in collaboration with Meyers (ref), and it's prototypes were used in COVID stuff (refs). However, `Episimlab` is designed to be used by anyone developing compartmental disease models. The package is useful for students, since it provides a minimal, approachable boilerplate for developing basic models in pure Python. At the same time, it introduces and reinforces best practices in object-oriented software development, such as modularity and reproducibility.

For disease modeling experts, `Episimlab` provides a platform that supports a wide variety of modeling use cases. It leverages concurrency in `xarray-simlab`, dataset chunking in `Dask`, and accelerated matrix math in `xarray`, so `Episimlab` models are performant even when using large (GB?) datasets. For example, Safegraph stuff (ref).

References