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

Computational models play a critical role in our scientific understanding of, and preparedness for, infectious diseases. These in silico disease models simulate real-world transmission dynamics, and are thereby well-suited for such tasks as early detection of novel pandemics, improving situational awareness during periods of high prevalence, and estimating efficacy of intervention strategies. Modeling approaches have proved valuable when responding to emergent epidemics such as the H1N1 flu pandemic, the Ebola epidemic, the Zika virus pandemic, and the recent COVID-19 pandemic (ref). For example, during the COVID-19 pandemic, compartmental disease models were instrumental for estimating future case counts and hospitalizations (refs). As more data on case incidence, hospitalizations, and viral genomics became available, disease modelers were able to incorporate new data streams to improve the performance of these compartmental models. These complex models are often expensive to develop, and few implementations of compartmental models shared a common software framework or application flow. ALSO MAKES IT DIFFICULT TO REPRODUCE, DISTRIBUTE, AND COMPARE DIFFERENT MODELS. Since EXPERTS EXPECT MORE PANDEMICS (ref), WE NEED TO SOLVE THIS PROBLEM.

Episimlab is a Python package that seeks to address this problem by providing a flexible framework for developing compartmental disease models. Models in Episimlab are collections of modular components, known as `processes`, which can be added, removed, or replaced to modify the dynamics of the simulated disease. In practice, this means that Episimlab supports development of models that:

1. Have many parameters, often with multiple, varying dimensions
2. Use any compartment structure that can be represented as a graph
3. Simulate interventions dynamically, such as administering vaccines only when case incidence exceeds a threshold.

The package is designed to be approachable; it includes several pre-packaged models that the user can run in a few lines of code. Commonly used `processes` such as calculating the force of infection (FOI) ship with Episimlab, and can be modified using class inheritance in Python. When the user chooses to add data streams or more complex transmission dynamics, they can easily do so by adding or replacing `processes` in the model. Episimlab also provides a scalable and performant runtime for model execution, thanks to integration with packages in the PyData stack such as Dask (Dask Development Team, 2016), Xarray (Hoyer & Hamman, 2017), and `xarray-simlab` (Bovy et al., 2021). Finally, Episimlab provides a standard for packaging, versioning, and sharing models, using Python's built-in class attributes.

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

Bovy, B., McBain, G. D., Gailleton, B., & Lange, R. (2021). *Benbovy/xarray-simlab: 0.5.0* (Version 0.5.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.4469813

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.148