CSDMS 2021 Annual Meeting

Git good: FAIR enough practices for research software

CoMSES NETWORK | OpenABM

CSDMS community surface dynamics modeling system

Arizona State University

University of Colorado Boulder

What we will cover today

1 FAIR principles for research software
2 Examples from the field
3 Break
4 Git to work / code along
5 Recap

FAIR principles for Research Software

License your code

https://choosealicense.com/

Document your code

narrative documentation and descriptive metadata

explicit, versioned, software, system, and data dependencies

community / domain specific standards

Deposit your code

TRUSTed digital repository or software registry

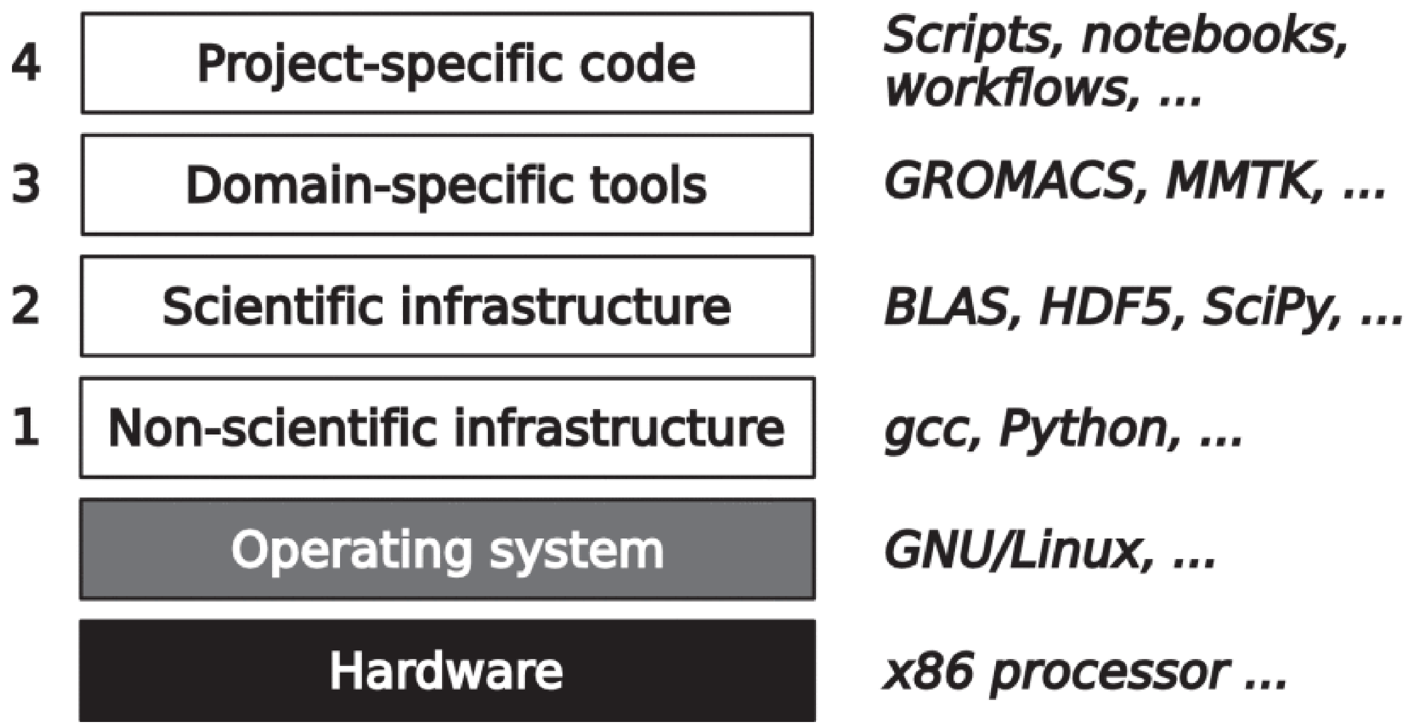
Use version control **← the hard part: what we'll cover today**

Software Collapse

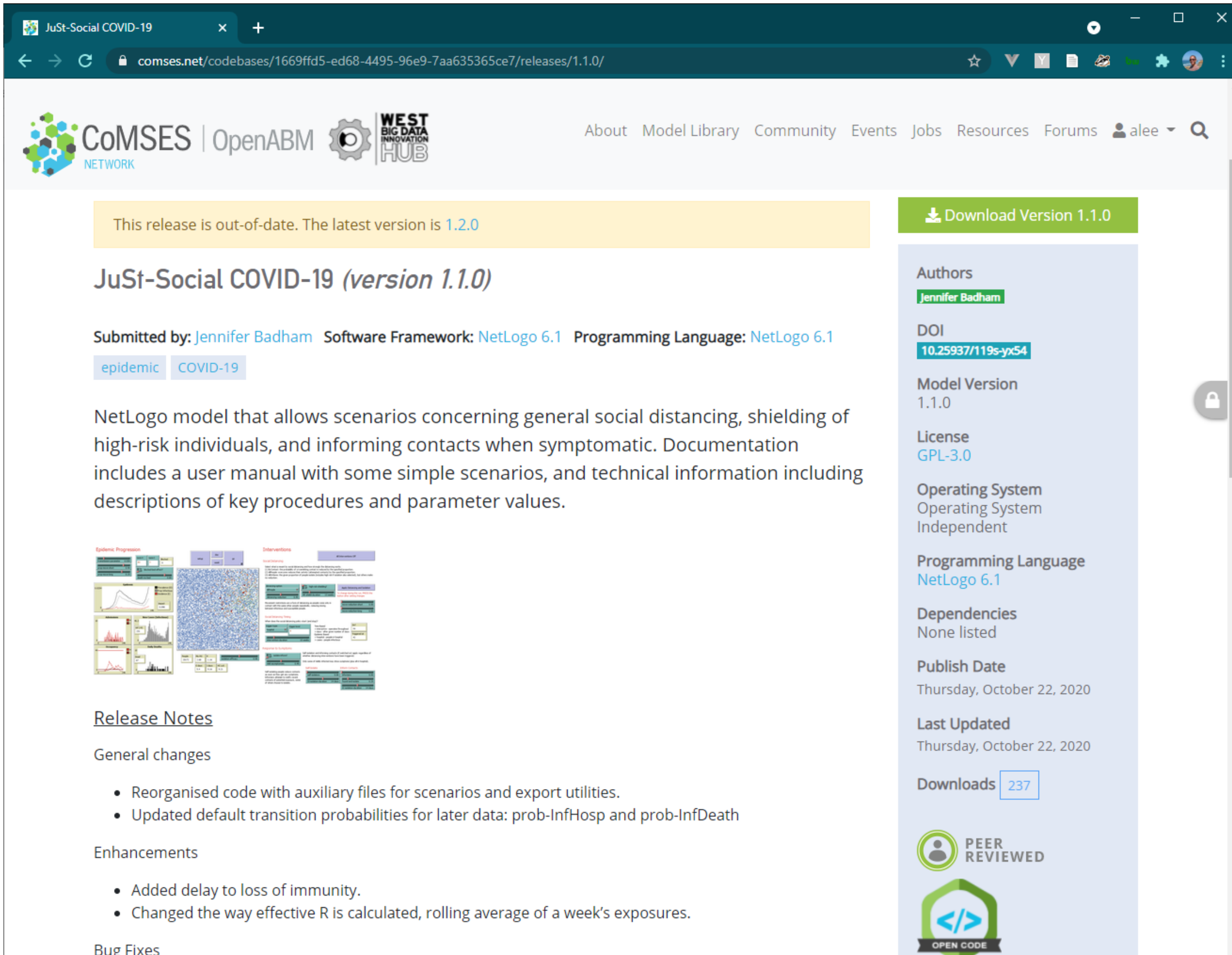
From https://en.wikipedia.org/wiki/File:Crooked_house_dudley.jpg

Mitigate Software Collapse

K. Hinsen, "Dealing With Software Collapse," in *Computing in Science & Engineering*, https://doi.org/10.1109/MCSE.2019.2900945



Examples


JuSt-Social COVID-19
comses.net/codebases/1669ffd5-ed68-4495-96e9-7aa635365ce7/releases/1.1.0/
CoMSES NETWORK
OpenABM
WEST BIG DATA INNOVATION HUB
About
Model Library
Community
Events
Jobs
Resources
Forums
alee
This release is out-of-date. The latest version is 1.2.0
Download Version 1.1.0
JuSt-Social COVID-19 (version 1.1.0)
Submitted by: Jennifer Badham Software Framework: NetLogo 6.1 Programming Language: NetLogo 6.1
epidemic
COVID-19
NetLogo model that allows scenarios concerning general social distancing, shielding of high-risk individuals, and informing contacts when symptomatic. Documentation includes a user manual with some simple scenarios, and technical information including descriptions of key procedures and parameter values.
Release Notes
General changes
Reorganised code with auxiliary files for scenarios and export utilities.
Updated default transition probabilities for later data: prob-InfHosp and prob-InfDeath
Enhancements
Added delay to loss of immunity.
Changed the way effective R is calculated, rolling average of a week's exposures.
Bug Fixes
Authors
Jennifer Badham
DOI
10.25937/119s-yx54
Model Version
1.1.0
License
GPL-3.0
Operating System
Operating System Independent
Programming Language
NetLogo 6.1
Dependencies
None listed
Publish Date
Thursday, October 22, 2020
Last Updated
Thursday, October 22, 2020
Downloads
237
PEER REVIEWED
OPEN CODE

Git work

- Fork the https://github.com/comses/git-fair-clinic repository
- Create a new branch with a proposed change that adds your name to the new CONTRIBUTORS.md file
- Create a pull request that adds your name and 1 other interesting fact to the CONTRIBUTORS.md file

Good Enough Practices for Git

Always pull (synchronize) your repository before you *commit* and *push* your changes

Commit early, commit often and strive for small, isolated changesets

Write descriptive commit logs that document the intent of your change

Resolve conflicts in the file marked by **<<<<, ====, >>>>**

Use branches for work in progress and keep your main branch stable

Use ***tags*** and ***releases*** to keep track of milestones (and connect Zenodo for automagic archival)

Maintain clarity and context in your repository's commit history with meaningful commit messages and linkages between commits, issues, and pull requests

Code Management Practices

Work towards **future-you-friendly code:**

1. Clearly commented with **assumptions laid bare**
2. **Self-documenting names for variables** that capture **semantics and purpose**
3. **Relative paths** for all input and output
4. **Test suite** with **sample inputs** and **expected outputs** and cover **edge cases**

Strive for **simplicity** and **modularity**

Encapsulate complexity with **clean interfaces/APIs**

Make **dependencies** explicit

Explain **parameters**

Have accompanying **narrative documentation** at **multiple levels of abstraction**

Adopt a **standardized directory structure**

```
Dockerfile
README
LICENSE
CITATION
data/
docs/
results/
src/
```

Testing

- Defines *contracts* on parts of your code.
- Given a set of inputs, what are the expected outputs within some ε
- For certain classes of scientific software, including simulation modeling, this is quite a bit harder.
- Consider which parts of your code can be extracted into smaller procedures and what assertions can be made about those procedures.

What's the difference between *Verification* and *Validation?*

https://software-carpentry.org/blog/2010/09/testing-scientific-software.html

https://software-carpentry.org/blog/2014/10/why-we-dont-teach-testing.html

Data Management Practices

Save raw and intermediate forms

In durable, open and non-proprietary machine-readable formats

Use meaningful names for files, directories, and columns/variables

Record all steps used to process data, ideally automated in a script

Describe your computational pipeline:

  Model -> Model output -> data processing and analysis -> results & visualizations

Analysis friendly data

site	1999	2000
Whitehorse	745	2666
Yellowknife	37737	80488
Inuvik	212258	213766

site	year	cases
Whitehorse	1999	745
Whitehorse	2000	2666
Yellowknife	1999	37737
Yellowknife	2000	80488
Inuvik	1999	212258
Inuvik	2000	213766

Wilson et al., 2016

TLDL Recap

Disciplined version control: small self-contained commits, descriptive log messages

Develop Future-You-Friendly codebases: make the time to refactor, coding ~ gardening ~ cleaning. Periodic maintenance can save time in the long run.

Adopt or develop community standards for documenting and structuring your code

Document your computational artifacts and pipelines: 50km and 5km

Archive your code and data in TRUSTed repositories and **cite code and data**

Use continuous integration: test your code in fresh environments regularly to detect impending **software collapse**