Benefits of Open Research in Social Simulation: An Early-Career Researcher's Perspective

Hyesop Shin (PhD)
Research Associate at the School of Geographical and Earth Sciences
University of Glasgow, UK

In March 2017, in my first year of PhD, I attended a talk at the Microsoft Research Lab in Cambridge UK. It was about the importance of reproducibility and replicability in science. Inspired by the talk, I redesigned my research beyond my word processer and hard disk to open repositories and social media. Through my experience, there have been some challenges to learn other people's work and replicate them to my project, but I found it more beneficial to share my problem and solutions for other people who may have encountered the same problem.

Having spoken to many early career researchers (ECRs) regarding the need for open science, specifically whether sharing codes is essential, the consensus was that it was not an essential component for their degree. A few answered that they were too embarrassed to share their codes online because their codes were not well coded enough. I somewhat empathised with their opinions, but at the same time, would insist that open research can gain more benefits than shame.

I wrote this short piece to openly discuss the benefits of conducting open research and suggest some points that ECRs should keep in mind. During the writing, there are some screenshots taken from my PhD work (Shin, 2021). I conclude my writing by accommodating personal experiences or other thoughts that might give more insights to the audience.

Benefits of Aiming an Open Project

I argue here that being transparent and honest about your model development strengthens the credibility of the research. In doing so, my thesis shared the original data, the scripts with annotations that are downloadable and executable, and wiki pages to summarise the outcomes and interpretations (see Figure 1 for examples). This evidence enables scholars and technicians to visit the repository if they are interested in the source codes or outcomes. Also, people can comment if any errors or bugs are identified, or the model is not executing on their machine or may suggest alternative ways to tackle the same problem. Even during the development, many developers share their work via online repositories (e.g. Github, Gitlab), and social media to ask for advice. Agent-based models are mostly uploaded on comses.net (previously named OpenABM). All of this can improve the quality of research.

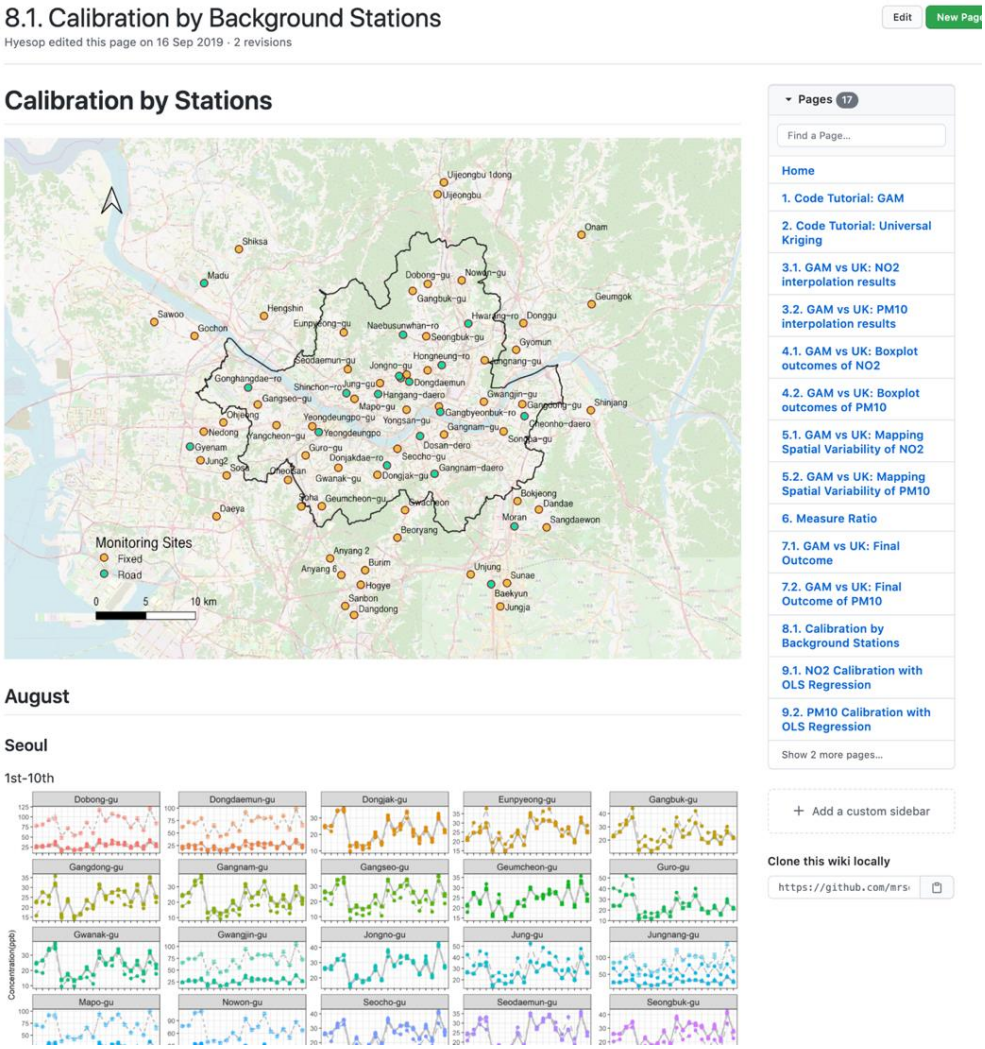

Figure 1 A screenshot of a Github page showing how open platforms can help other people to understand the outcomes step by step

More practically, one can learn new ideas by helping each other. If there was a technical issue that can't be solved, the problem should not be kept hidden, but rather be opened and solved together with experts online and offline. Figure 2 is a pragmatic example of posing questions to a wide range of developers on Stackoverflow – an online community of programmers to share and build codes. Providing my NetLogo codes, I asked how to send an agent group from one location to the other. The anonymous person, whose ID was JenB, kindly responded to me with a new set of codes, which helped me structure the codes more effectively.

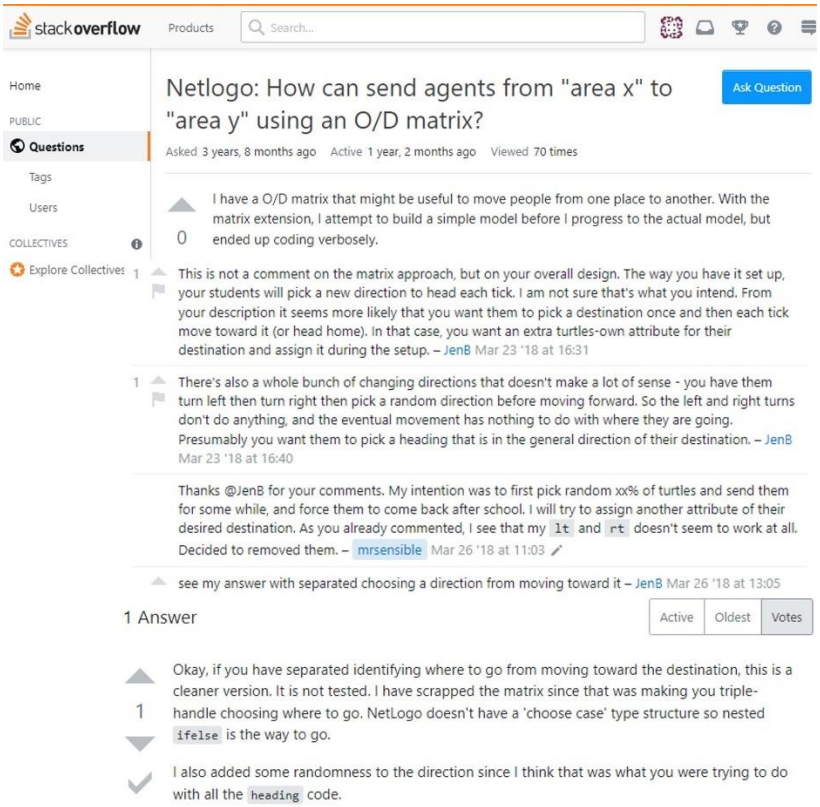

Figure 2 Raising a question about sending agents from one location to another in NetLogo

Another example was about the errors I had encountered whilst I was running NetLogo with an R package "nlrx" (Salecker et al., 2019). Here, R was used as a compiler to submit iterative NetLogo jobs on the HPC (High Performance Computing) cluster to improve the execution speed. However, much to my surprise, I received error messages due to early terminations of failed HPC jobs. Not knowing what to do, I posed a question to the developer of the package (see Figure 3) and luckily got a response that the R ecosystem stores all the assigned objects in the RAM, but even with gigabytes of RAM, it struggles to write 96,822 patches over 8764 ticks on a spreadsheet.

Stackoverflow has kindly informed that NetLogo has a memory ceiling of 1GB[1] and keeps each run in the memory before it shuts down. Thus, if the model is huge and requires several iterations, then it is more likely that the execution speed will decrease after a few iterations. Before this information was seen, it was not understood why the model took 1 hour 20 minutes to finish the first run but struggled to maintain that speed on the twentieth run. Hence, sharing technical obstacles that occur in the middle of research can save a lot of time even for those who are contemplating similar research.

[1] http://ccl.northwestern.edu/netlogo/docs/faq.html#how-big-can-my-model-be-how-many-turtles-patches-procedures-buttons-and-so-on-can-my-model-contain

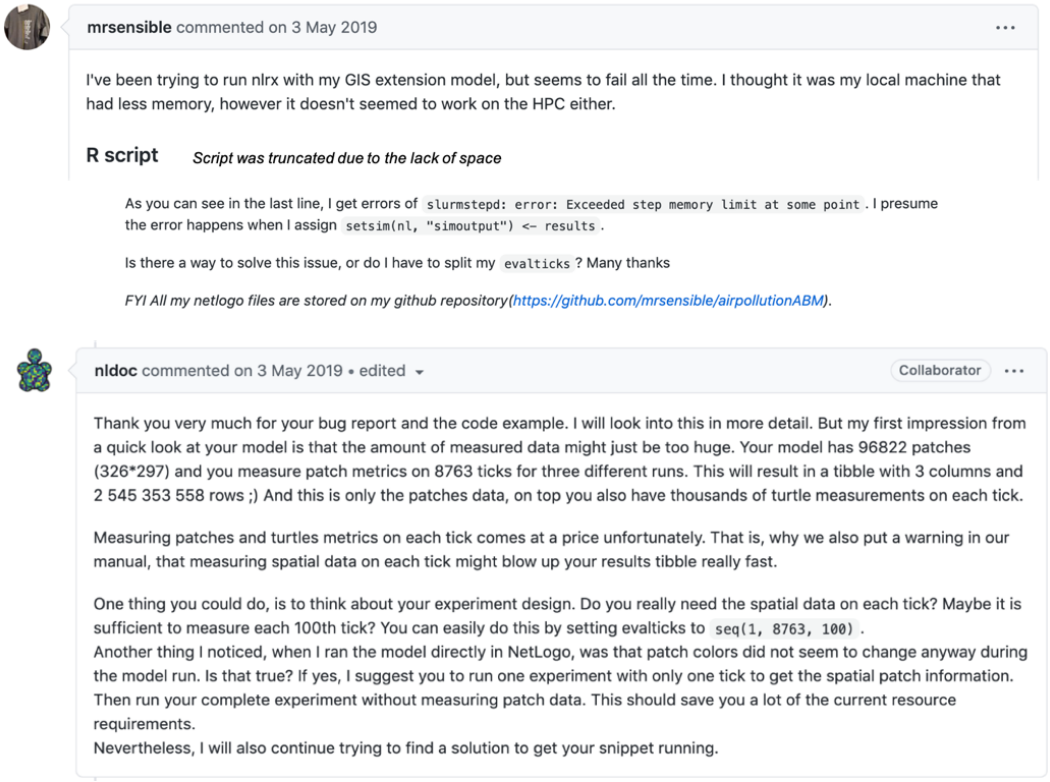

Figure 3 Comments posted on an online repository regarding the memory issue that NetLogo and R encountered

The Future for Open Research

For future quantitative studies in social simulation, this paper suggests students and researchers in their early careers should acclimatise themselves to using open-source platforms to conduct sustainable research. As clarity, conciseness, and coherence are featured as the important C's for writing skills, good programming should take into consideration the following points.

First is clarity and conciseness (C&C). Here, clarity means that the scripts should be neatly documented. The computer does not know whether the codes are dirty or neat, it only cares whether it is syntactically correct, but it matters when other people attempt to understand the task. If the outcome produces the same results, it is always better to write clearer and simpler codes for other people and future upgrades. Thus, researchers should refer to other people's work and learn how to code effectively. Another way to maintain clarity in coding is to keep descriptive and distinctive names for new variables. This statement might seem contradictory to the conciseness issue, but this is important as one of the common mistakes users make is to assign variables with abstract names such as LP1, LP2…LP10, which seems clear and concise for the model builder, but is even harder for the others when reviewing the code. The famous quote from Einstein, "Everything should be made as simple as possible, but not simpler." is the appropriate phrase that model builders should always keep in mind. Hence, instead of coding LP9, names such as LandPriceIncreaseRate2009 (camel cases) or landprice_incrate_2009 (snake cases) can be more effective for the reviewers to understand the model.

Second is reproducibility and replicability (R&R). To be reproducible and replicable, initially, no errors should occur when others execute the script, and possible errors or bugs should be reported. It will also be more useful to document the libraries and the dependencies required. This is quite important as different

OSs (operating systems) have different behaviours to install packages. For instance, the sf package in R has slightly different ways to install the package between OSs where Windows and MacOSX can be installed from the binary package while Linux needs to separately install GDAL (to read and write vector and raster data), Proj (which deals with projection), and GEOS (which provides geospatial functions) prior to the package installation. Finally, it would be very helpful if unit testing is included in the model. While R and Python provide splendid examples in their vignettes, NetLogo remains to offer the library models but goes no further than that. Offering unit testing examples can give a better understanding when the whole model is too complicated for others to comprehend. It can also give the impression that the modeller has full control of the model because without the unit test the verification process becomes error-prone. The good news is that NetLogo has most recently released the Beginner's Interactive Dictionary with friendly explanations with videos and code examples[2].

Third is to maintain version control. In terms of sustainability, researchers should be aware of software maintenance. Much programming software relies on libraries and packages that are built on a particular version. If the software is upgraded and no longer accepts the previous versions, then the package developers need to keep updating to run it on a new version. For example, *NetLogo 6.0* experienced a significant change compared to versions 5.X. The biggest change was the replacement of tasks[3] by anonymous procedures (Wilensky, 1999). This means that tasks are no longer primitives but are converted to arrow syntax. For example, if there is a list of [a b c], the previous task is asked to add the first, second, and third element as foreach [a b c] [?a+?b+?c], while the new version does the same job as foreach [a b c][add_all → a + b + c]. If the models haven't converted to a new version it can be viewable as a read-only model but can't be executed. Other geospatial packages in R such as *rgdal* and *sf*, have also struggled whenever a major update was made on their own packages or on the R version itself due to a lot of dependencies. Even ArcGIS, a UI (User Interface) software, had issues when they upgraded it from version 9.3 to 10. The projects that were coded under the VBA script in 9.3 were broken because it was not recognised as a correct function in the new version based on Python. This is also another example that backward compatibility and deprecation mechanisms are important.

Lastly, for more advanced users, it is also recommended to use a collaborative platform that executes every result from the codes with the exact version. One of the platforms is Codeocean (https://codeocean.com/). The Nature research team has recently chosen the platform to peer-review the codes (Perkel, 2019). The Nature editors and peer-reviewers strongly believed that coding has become a norm across many disciplines, and hence have asserted that the model process including the quality of data, conciseness, reproducibility, and documentation of the model should be placed as a requirement. Although the training procedure can be difficult at first, it will lead researchers to conduct themselves with more responsibility.

Looking for Opinions

With the advent of the era of big data and data science where people collaborate online and the 'sharing is caring' atmosphere has become a norm (Arribas-Bel et al., 2021; Lovelace, 2021), I insist that open research should no longer be an option. However, one may argue that although open research is by far an excellent model that can benefit many of today's projects, there are certain types of risks that might concern ECRs such as intellectual property issues, code quality and technical security. Thus, if you have had different

[2] https://ccl.northwestern.edu/netlogo/bind/
[3] Tasks can be equations, x + y, or a set of lists [1 2 3 4 5]

opinions regarding this issue, or simply favour adding your experiences during your PhD in social simulation, please add your thoughts via a thread.

References

Arribas-Bel, D., Alvanides, S., Batty, M., Crooks, A., See, L., & Wolf, L. (2021). Urban data/code: A new EP-B section. *Environment and Planning B: Urban Analytics and City Science*, 23998083211059670. https://doi.org/10.1177/23998083211059670

Lovelace, R. (2021). Open source tools for geographic analysis in transport planning. *Journal of Geographical Systems*, *23*(4), 547–578. https://doi.org/10.1007/s10109-020-00342-2

Perkel, J. M. (2019). Make code accessible with these cloud services. *Nature*, *575*(7781), 247.

Salecker, J., Sciaini, M., Meyer, K. M., & Wiegand, K. (2019). The nlrx r package: A next-generation framework for reproducible NetLogo model analyses. *Methods in Ecology and Evolution*, *10*(11), 1854–1863. https://doi.org/10.1111/2041-210X.13286

Shin, H. (2021). *Assessing Health Vulnerability to Air Pollution in Seoul Using an Agent-Based Simulation*. University of Cambridge. https://doi.org/https://doi.org/10.17863/CAM.65615

Wilensky, U. (1999). *Netlogo*. Northwestern University: Evanston, IL, USA.