Lessons learned replicating the analysis of outputs from a social simulation of biodiversity incentivisation

Gary Polhill [a], Lorenzo Milazzo [a,b], Terry Dawson [b], Alessandro Gimona [a] and Dawn Parker [c]

[a] The James Hutton Institute, Aberdeen, UK
{gary.polhill; lorenzo.milazzo; alessandro.gimona}@hutton.ac.uk
[b] The University of Dundee, Dundee, UK
t.p.dawson@dundee.ac.uk
[c] The University of Waterloo, Waterloo, Canada
dcparker@uwaterloo.ca

Abstract. This paper reports on an exercise in replicating the analysis of outputs from 20,000 runs of a social simulation of biodiversity incentivisation (FEARLUS-SPOMM) as part of the MIRACLE project. Typically, replication refers to reconstructing the model used to generate the output from the description thereof, but for larger-scale studies, the output analysis itself may be difficult to replicate even when given the original output files. Tools for analysing simulation output data do not facilitate keeping records of what can be a lengthy and complicated process. We provide an outline design for a tool to address this issue, and make some recommendations based on the experience with this exercise.

Keywords: social simulation outputs; metadata; analysis

1 Introduction

Replication of social simulation results has been highlighted as a significant issue for the discipline for a number of years (e.g. Edmonds and Hales 2003), and has even led to a short series of workshops (Hales et al. 2003; Rouchier et al. 2008). The focus of replication work has thus far been on the model itself, but as will be shown here, the analysis of the outputs of the model can potentially be just as complex, and no less difficult to replicate unless adequate records are kept. Schmolke et al.'s (2010) TRACE protocol provides some guidance highlighting the need to keep a notebook of the analysis done; however, lessons can be learned from the replication exercise reported herein that provide more detailed guidance on the information that should be recorded, and on tools that could be used to support the process.

The output analysis replication in this paper concerns earlier work with FEARLUS-SPOMM, which is a coupled agent-based model of agricultural decision-making and species stochastic patch occupancy metacommunity model that has been used to explore incentivisation strategies to improve biodiversity (Polhill et al. 2013; Gimona and Polhill 2011). Belonging to the 'typification' class of social simulations

("theoretical constructs intended to investigate some properties that apply to a wide range of empirical phenomena that share some common features" – Boero and Squazzoni, 2005), recent work involved the analysis of the outputs from around 20,000 runs of the model using a number of techniques aimed at demonstrating nonlinearities in the relationship between incentivisation and biodiversity outcome.

Recording workflow data on the process used to create analysis can be challenging, and currently there are no codified standards as to how this should be done for ABMs. For FEARLUS-SPOMM, the methods used drew heavily on statistical techniques available as R packages that are as part of core R functionality. Although R allows transcripts of interactive terminal sessions to be saved, the work involved great deal of exploration of different ways of attempting to visualise and analyse the nonlinearities in the model results, not all of which were likely to be reported in the paper. Such logs are therefore not the best way to record the means by which the outputs were analysed, and hence the strategy used was to save each analysis or visualisation in a(n R) script. Since the output from the (Swarm) software that generated the output data being analysed used a mixture of text formats, some Perl scripts were also written to process that output into a CSV file for easy import into R. When the MIRACLE project (Parker et al. 2015) provided a context in which the replication of that analysis was necessary, an opportunity was created to test the viability of the above strategy.

In the rest of this paper, we describe the information available for the output analysis replication process, how it was used to regenerate some of the figures in Polhill et al. (2013) (and the information that was missing that would have facilitated this process), and we describe a prototype tool to support keeping the records needed to perform replication more easily, before making some recommendations in concluding remarks.

2 Replicating the output analysis

This section briefly summarises the experiments done in Polhill et al. (2013), before describing the way in which the analysis was reconstructed from the information available.

2.1 Summary of the analysis and techniques used

The original work conducted a series of simulations exploring the parameter space of the model, with a particular focus on the relationship between the amount of incentive to farmers offered by each mechanism and the resulting landscape-scale species richness after 200 iterations of the model. Four incentivisation mechanisms were explored covering the following two dimensions; each has as parameter an incentive amount offered:

- Providing incentives for specific activities aimed at improving biodiversity versus providing incentives for biodiversity outcomes.

- Providing incentives to farmers individually versus providing increased incentives to farmers when they and their neighbours delivered the activity or outcome the mechanism required (referred to as 'clustered' incentives').

To cover other circumstantial factors, the model explored two levels of aspiration for profit in the farming agents, two levels of input costs, and two stylised time series of the market prices for the goods produced on the farm. The combination of incentivisation mechanism, market, input costs and aspirations is referred to as a 'scenario'; there are 32 of these.

Incentive values covering 1, 2, …, 10 were explored, with 20 runs per parameterisation. This makes 6400 runs. Differences between the incentivisation mechanisms mean that some naturally spend more money than others given the same set of circumstances. To correct for this, the 6400 runs were repeated twice further, once dividing the clustered incentive values by 2 (i.e. 0.5, 1, 1.5, … 5 instead of 1, …, 10), and once dividing these by 10 (i.e. 0.1, …, 1). This makes 19200 runs. In a second phase of runs, the non-clustered incentivisation mechanisms were explored using incentive values 15, 20, 25, 30, 40, 50, 100, requiring a further 2240 runs.

Analysis of the outputs reported in the paper covered the following:

- Rejecting runs with levels of expenditure that are too high, or levels of bankruptcy that are too high. These runs were deemed unrealistic, and the rejection left 16949 runs for use in subsequent stages.
- Testing for nonlinearity in each scenario in the relationship between incentive amount and landscape scale species richness in the last recorded time step of the model. Five tests were used, the details of which are not important here (see Polhill et al. (2013) for details), but they involved building and comparing Generalised Additive Models (GAMs; Hastie and Tibshirani, 1990) and linear models of the relationship, as well as computing the Akaike Information Criterion (Akaike 1973).
- Using recursively partitioning classification trees (Breiman et al. 1984) to see how scenario variables and expenditure combined to deliver landscape-scale species richness.

Archives of the raw simulation output files were kept. The goal of the exercise reported here is for another team member not involved in the original research to replicate the output analysis described above given these files.

2.2 Records kept

The following details the available information for the reconstruction process:

- diaries of research activities on analysing the outputs starting 19 October 2011 and running until 10 August 2012;
- Perl, shell and R scripts that were intended to act as documentation of these activities;
- a README file describing some of the Perl and R scripts, and associated output files.

The diaries were contained principally in two files covering consecutive time periods, but were also distributed in different versions across different computers used to perform the analysis. The same applied to the scripts, of which there are 74 R scripts 29 Perl scripts and 19 shell scripts. However, these scripts can be grouped as several of them are updated versions of others, fixing bugs or providing additional or variant functionality. As scripts essentially aimed at recording an analysis or visualisation that would otherwise be done interactively with the R console, and hence less 'formal' computer software, version control software was not used to keep track of these updates. More importantly, several approaches to analysing and visualising the output were tried during the course of the work, most of which were 'dead ends' or not included in the final paper for other reasons. Although the README file and diaries provided some guidance, they were not always sufficient to record precisely which version of which script was used to produce a particular result.

2.3 Reconstructing the output analysis

Figure 1 shows the process by which the various analysis and visualisation artefacts in the paper (shaded nodes) were produced. Each run of the model generates a number of files. The SPOMM half creates several CSV files storing data about species occupancy in various different ways. The FEARLUS half has a bespoke text output format for reports on the model, and another output format loosely based on ARC's Grid/ASCII format for reporting on spatial data in which multiple layers of spatial data for exactly the same region and division of space are saved in the same file.

The first task in analysing the output was to create a single CSV file containing salient information on each run. This involved processing the several thousand files from all the runs of FEARLUS-SPOMM using a Perl script that provided a summary of each run, one per line of the CSV file. There were six versions of this script on two different machines, four of which contained bugs fixed in later versions, and a fifth contained functionality used for a later set of runs. Although this should mean that it is trivial to deduce which version of the script was used to create the CSV file used for subsequent analysis (the one that has the fixed bugs but not the additional functionality), some of the R scripts associated with earlier analysis contained corrections to the data to address the bugs that were fixed later. Although the scripts used in the published analysis did not contain these corrections, it would have been more reassuring to have had a record of which version was used to create the CSV file used by these scripts, and indeed of which CSV file these scripts used.

As mentioned earlier, some runs reported in the resulting CSV file were eliminated from consideration because of overly high bankruptcy or expenditure rates. This was done using the R console, reading the file in, removing the offending runs from the data frame, and saving the result as a new CSV file. However, during the reconstruction process, it became clear that subsequent scripts assumed the existence of columns in the data that were not present in the output from the processing script. The code to produce these columns was eventually found in an R library that had been created because this functionality was being replicated in a number of the analysis scripts used during exploratory analysis.

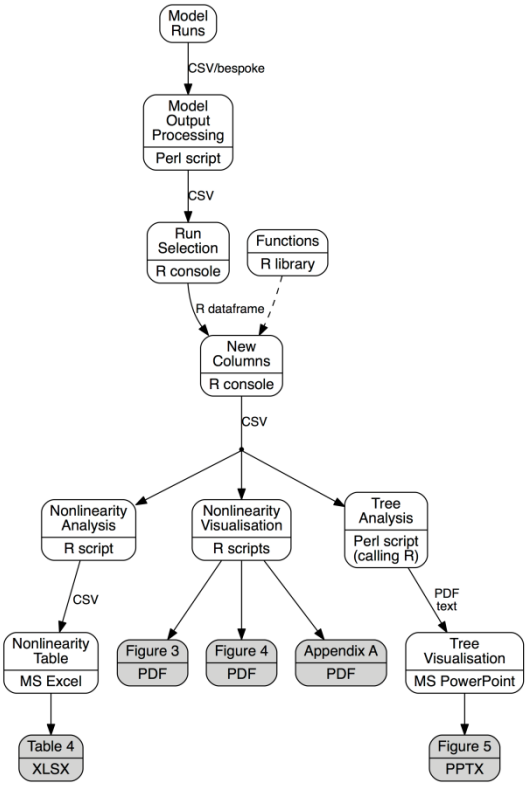

Fig. 1. Workflow of the output analysis process

The remaining scripts cover different visualisations and analyses reported on in the paper, and feature similar issues to the output file processing script discussed above. The analysis of nonlinearity was done using a script with five versions, and visualised using a script with four; however, differences in the layout and style of the visualisations in various figures in the Polhill et al. (2013) paper meant that the scripts used for these were more definite. The five versions of the nonlinearity analysis scripts consist of an original version of the script, and four revisions of it that variously change default settings to those eventually used in the paper, and add some functionality not used. This also meant that it was possible to deduce which version of the script had been deployed. The output of this script (another CSV file) was subsequently loaded into a spreadsheet, which was used to generate the table used in the paper. The decision tree analysis was conducted with a Perl script that calls R while it is running. This produces a PDF containing graphs showing distributions of key model variables at each leaf node of the tree, and text describing the tree structure as part of the output to the terminal. The diagram used in the paper was constructed by hand from these

files using presentation software, with the ‘notes’ section of the slide containing the diagram consisting of text pasted from the terminal showing the command used and the output therefrom. Hence, although there were three versions of the script to produce the tree, the version used was known, as were the command-line options, which, since they include configuration parameters for the decision tree algorithm (rpart()), are critical for successful reconstruction of the output.

3 Designing tools to support output analysis

The missing information that would have critically helped in reproducing the output analysis described above is provenance-based – which script used which file(s) as input and generated which other file(s) as output, though other practices, discussed later, would also have facilitated this process. Identification of the key processes (events, actions) is one the crucial points that needs to be addressed in order to guarantee the replication (reproducibility) of any experiment or study. In the case of an in-silico experiment, events and actions are associated with running applications, scripts and other software. At the most abstract level, the applications are making use of various forms of ‘container’; which in specific cases could be files (e.g. input/output files) or data structures (e.g. arrays or hash tables containing initialization or input data).

In general, an off-the-shelf Workflow Management System (e.g. Taverna – Wolstencroft et al. 2013) can be adopted to implement a (semi-)automated system to run the simulation, process and analyse the data. However, we propose a ‘light-weight’ approach with a view to minimising the learning curve associated with a fully-fledged application, and decided to develop a bespoke application written in Python that also allows the replication of an entire study. Assuming a working practice in which scripts (e.g. in R) are written to cover each stage of the analysis process, the Python application is intended to ‘wrap’ calls to the script, thereby keeping track of the required provenance.

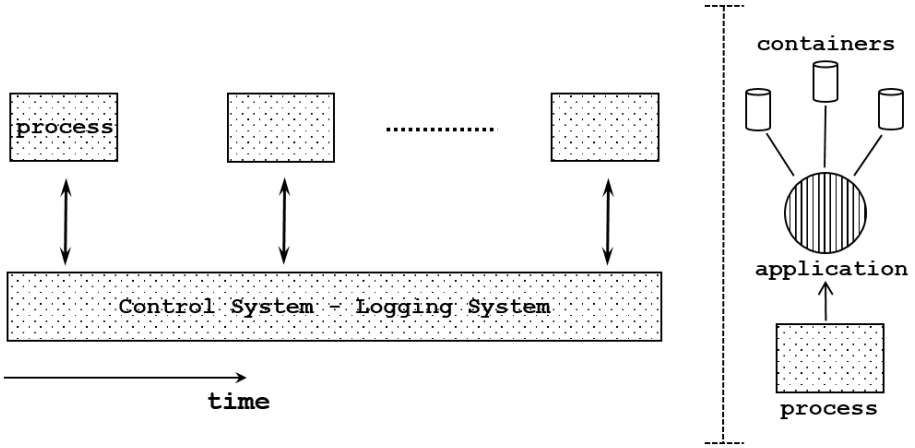

Fig. 2. Schematic diagram of the software system

The current version of the software consists of several components (figure 2) including: a) a procedure for 'forking' and monitoring processes, b) a logging system c) procedures to process different kinds of data (I/O, metadata), d) an interface allowing the storage/retrieval of data into/from a relational database.

The logging system plays a crucial role in the system. A logging system has been implemented to perform bookkeeping, scheduling, and error handling, and includes capture of provenance metadata. The monitoring carried out by the logging system is closely related to the identification, organization, and storage of the relevant information describing the simulation study. These key data are processed according to a set of pre-established metadata for social simulation outputs, which is the subject of on-going work.

4 Recommendations and future work

At least one of the stages in producing the final results could have been eliminated if the model had not produced output data in a bespoke text format. (Indeed, later versions of FEARLUS use XML rather than the bespoke text format for the runs reported here.) At last year's Social Simulation Conference in Barcelona, participants were asked to provide a record of their current practice in recording simulation outputs and associated metadata. The results (figure 3) show a strong preference for CSV formats (with or without a header line describing the data in the columns). Several people also use a tab-separated text format.

The reasons for this are obvious: CSV and tab-separated formats are almost universally readable by other software that might be used to do analysis of the outputs (e.g. R and spreadsheet software). Indeed, Repast Simphony and NetLogo (arguably the two most popularly-used agent-based modelling toolkits) use CSV format (or have an option to do so) in their parameter sweep and behaviour space (to use their respective nomenclature) batch run tools.

One important recommendation therefore is in encouraging developers of agent-based models to use output formats that are readily used by other software. CSV and tab-separated formats may not always be suitable for encoding outputs from social simulations, however. In particular, although CSV files can be used for raster spatial data using columns for *x* and *y* co-ordinates, vector-based representations may be less easy to encode this way, especially if the model makes important distinctions between point, line, area and volume spatial regions and the properties they have. Output data may also need to encode properties of different classes of object more generally. This suggests that broader support is needed for data formats such as XML or JSON that allow some encoding of the structure of the data. Capturing the relational structure of the output may also suggest adopting a practice of saving outputs in relational databases (though this was one of the least popular options in the sample). Tools such as sqlite3 offer options for providing such storage without the investment in configuration and management of larger RDBMS systems such as MySQL and PostgreSQL. R

provides support both for reading XML files and for interacting with databases, but these tools are much less popularly used.

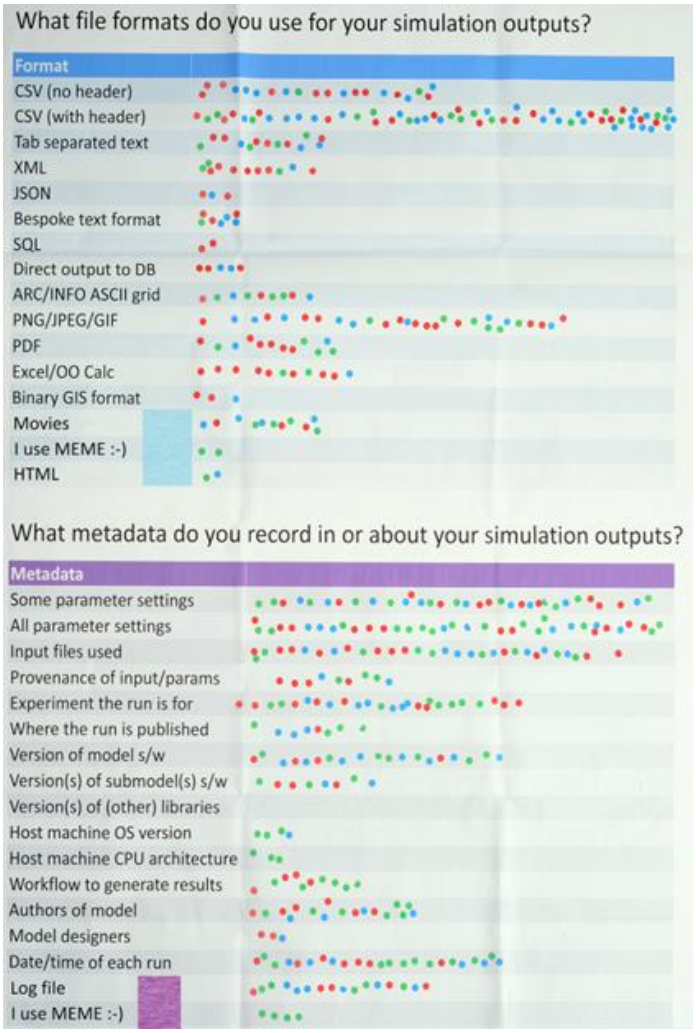

Fig. 3. Poster at last year's SSC capturing current practice. Entries added by participants are indicated with the blue/purple rectangle.

A second recommendation is that tools be provided that facilitate the recording of provenance metadata for the results of output analyses. Two important aspects of provenance would have helped with this reconstruction: (i) knowing which script was used to generate which output file, which input files were used, and which command-line options; (ii) knowing which script is a revision of which other script or scripts. Although recommendations such as PROV (Moreau and Missier 2013) provide the basis on which such information can be structured, popularly-used tools do not save

such metadata. Log files of console sessions (both for the shell terminal and for R when this was used interactively) would have captured the information needed to perform the reconstruction, but they potentially contain a great deal of noise when attempting to find out specific information about how a particular visualisation or result was produced. Similarly, the diary contains information not relevant to the paper, since it documents numbers of other analyses that were not included. Worse, as free text, it would be much more challenging to provide an automated search facility that could theoretically extract the information associated with a specific query.

Using scripts to perform particular analysis and visualisation tasks does at least act as a record of that task, and one that can theoretically be reused for other tasks. Although there were several scripts, it took a only few hours to deduce which scripts were associated with which table or figure in the paper; something that could have taken a lot longer if searching through log files of console sessions or (worse) if no such log files were available at all. Nevertheless, the temptation to add functionality using command line options and switches poses an obstacle to reconstructing output analysis if it is not clear which options were used when generating particular files of interest. One approach to addressing the problem is to capture the entire workflow in a shell script; however, this only works if the analysis can be done on a single machine, and if certain stages in the process take a long time (and do not need replication once executed once), running the whole script each time a change is made to the analysis process is an inefficient use of computer resources. Dealing with the problem by making several copies of scripts effectively fixing combinations of options that would otherwise have been implemented in a single script would not necessarily facilitate output analysis reconstruction, and would compound problems with code maintenance and bug fixing during analysis. Although here an R library was created with a view to allowing commonly-used functions to be reused from one script to the next, this is not necessarily recommended due to the overheads associated with maintaining R libraries (particularly in across R versions and operating systems).

For some researchers, there is a 'flow' associated with data analysis that is interrupted if best-practice procedures of faithfully recording all activities is rigorously followed. Although in the work reconstructed here, best-practice has arguably been followed through keeping a diary of the work done, saving analyses in documented scripts, and in some cases saving terminal transcripts, these have not between them been sufficient to make the reconstruction process trivial. Some detective work has been necessary, and there have been cases where steps in the analysis have not been documented. For those willing to invest in learning how to use them, a tool such as AITIA's MEME (Ivanyi et al., 2007) offers functionality that supports the analysis of batch runs of agent based models, and in this case, is licensed using a GNU GPL. The approach used in this paper has been to wrap scripts in another program that maintains provenance metadata.

Although facilities such as openabm.org are provided for archiving models, and journals are increasingly recommending the use of such archives, clearly there is an argument for recommending also archiving the processes by which the results were analysed and tables and diagrams in the paper generated.

Acknowledgements.

This work was funded by a number of agencies under the Digging into Data Challenge (Third Round) and by the Scottish Government Rural Affairs and the Environment Portfolio Strategic Research Theme 1 (Ecosystem Services). Computing facilities have been provided by Compute Canada and Sharcnet.

References.

1. Akaike, H. (1974) A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19** (6), 716-723.
2. Boero, R. and Squazzoni, F. (2005) Does empirical embeddedness matter? Methodological issues on agent-based models for analytical social science. *Journal of Artificial Societies and Social Simulation* **8** (4), 6. http://jasss.soc.surrey.ac.uk/8/4/6.html
3. Breiman, L., Friedman, K., Olshen, R. A. and Stone, C. J. (1984) *Classification and Regression Trees*. Boca Raton, Florida: Chapman & Hall/CRC.
4. Edmonds, B. and Hales, D. (2003) Replication, replication and replication: some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation* **6** (4), 11. http://jasss.soc.surrey.ac.uk/6/4/11.html
5. Gimona, A. and Polhill, J. G. (2011) Exploring robustness of biodiversity policy with a coupled metacommunity and agent-based model. *Journal of Land Use Science* **6** (2-3), 175-193.
6. Hales, D., Rouchier, J. and Edmonds, B. (2003) Model-to-model analysis. *Journal of Artificial Societies and Social Simulation* **6** (4), 5. http://jasss.soc.surrey.ac.uk/6/4/5.html
7. Hastie, T. J. and Tibshirani, R. J. (1990) *Generalized Additive Models*. Boca Raton, Florida: Chapman & Hall/CRC.
8. Ivanyi, M., Bocsi, R., Gulyas, L., Kozma, V. and Legendi, R. (2007) The Multi-Agent Simulation Suite. *22nd Conference on Artificial Intelligence (AAAI-07), Vancouver, British Colombia, 22-26 July 2007*.
9. Moreau, L. and Missier, P. (2013) (eds.) *PROV-DM: The PROV Data Model. W3C Recommendation 30 April 2013*. http://www.w3.org/TR/2013/REC-prov-dm-20130430/
10. Parker, D. C., Barton, NM., Dawson, T., Filatova, T., Jin, X., Lee, J.-S., Polhill, J. G., Robinson, K. and Voinov, A. (2015) The MIRACLE project: A community library to archive and document analysis methods for output from agent-based models. *Association of American Geographers 2015 Annual Meeting, April 21-25, 2015, Chicago, Illinois*.
11. Polhill, J. G., Gimona, A. and Gotts, N. M. (2013) Nonlinearities in biodiversity incentive schemes: A study using an integrated age0tbased and metacommunity model. *Environmental Modelling and Software* **45**, 74-91.
12. Roucher, J., Cioffi-Revilla, C., Polhill, J. G. and Takadama, K. (2008) Progress in model-to-model analysis. *Journal of Artificial Societies and Social Simulation* **11** (2), 8. http://jasss.soc.surrey.ac.uk/11/2/8.html
13. Schmolke, A., Thorbek, P., DeAngelis, D. L. and Grimm, V. (2010) Ecological models supporting environmental decision making: a strategy for the future. *Trends in Ecology and Evolution* **25** (8), 479-486.
14. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S. and Goble, C. (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* **41** (W1), W557-W561.