Software Impacts 21 (2024) 100689

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

RIGSS — Inverse Generative Social Science using R

Thomas Chesney [a,*], Robert Pasley [a], Muhammad Asif Jaffer [b]

[a] Nottingham University Business School, United Kingdom
[b] IBA Karachi - Institute of Business Administration, Pakistan

ARTICLE INFO

Keywords:
Genetic programming
Agent-based modelling
Social theory

ABSTRACT

We present RIGSS, software that can be used to run an Inverse Generative Social Science study in R. We give a brief overview of this research method and explain how our software can be used. We implement a Hawk Dove game as an executable example. We then discuss the potential that Inverse Generative Social Science has.

Code metadata

Current code version	*v1*
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2024-134
Permanent link to reproducible capsule	
Legal code license	*GNU General Public License (GPL)*
Code versioning system used	*none*
Software code languages, tools and services used	*R*
Compilation requirements, operating environments and dependencies	
If available, link to developer documentation/manual	
Support email for questions	robert.pasley@nottingham.ac.uk

1. Inverse generative social science

When used for social science, agent-based modelling creates a model of a target – the social situation being studied – that can be experimented on [1]. When the model is run, the agents interact with each other and with their environment by following rules that have been programmed by the modeller who then stands back and observes the outcome [2]. Inverse Generative Social Science [3] reverses this by using evolutionary computing [4] to evolve these rules by natural selection. Specifically our software uses genetic programming [5]. The system attempts to evolve an agent-based model whose output matches an ideal description of agent behaviour called the reference dataset. Typically this will take many days of computation on a normal computer.

RIGSS is an implementation of Inverse Generative Social Science in R. To use it, the 'skeleton' of the agent model must be created as an R function. The skeleton is the entire agent model minus the rule or rules that agents follow. The function accepts one parameter—*evolvedRule*—which is the current rule written in R code, generated by genetic programming. This rule is inserted into the agent model, which is then run. At the end of the run, the function returns data that is compared to a reference dataset in order to test that rule's fitness. This fitness score is then used to evolve a next generation of rules, as described in the following section.

2. Software overview

For a usable example, at the RIGSS GitHub page the file *Agent Model* implements a Hawk Dove game as an agent model function (see [6] for an explanation of Hawk Dove).

The file *RIGSS functions* contains the functions for creating the agents' rules. RIGSS stores each rule in a tree structure. This allows for the three genetic operations of genetic programming—crossover, mutation and reproduction. (The tree ensures that rules can be interpreted by R without error—otherwise brackets and other mathematical symbols might evolve in the wrong place.) Initially a generation of rules is created at random. Then the genetic operations are used to create successive generations of rules as they evolve to improve fit:

* Corresponding author.
E-mail addresses: thomas.chesney@nottingham.ac.uk (T. Chesney), robert.pasley@nottingham.ac.uk (R. Pasley), masif@iba.edu.pk (M. Asif Jaffer).

https://doi.org/10.1016/j.simpa.2024.100689
Received 24 June 2024; Received in revised form 22 July 2024; Accepted 22 July 2024

Reproduction: one of the rules (chosen with probability proportional to its fitness, so most likely a rule with high fitness) is allowed to reproduce and simply moves into the next generation.

Mutation: one of the rules (again one most likely that has high fitness) mutates randomly and the mutated rule moves into the next generation. A mutation means that part of the rule is changed at random.

Crossover: two of the rules (as before two that most likely have high fitness) breed and their two children rules move into the next generation. This breeding is known as crossover. This means that part of the first rule and part of the second rule are swapped with each other.

An example rule from the Hawk Dove model is:

IF A/ B > C THEN TAKE = 50

This rule represents the agent behaviour that is being evolved. This is clearly model specific. In Hawk Dove, the agents must decide how much of a resource to take; other models will have different rule structures and RIGSS must be edited by the user for their particular model. In this case, the commands "IF" and "THEN TAKE = " are set and do not change. Apart from "IF" and "THEN TAKE = ", all the other elements in the example rule above – A, divide, B, greater than, and 50 – are evolved.

In the *RIGSS functions* file, the first thing that is done is defining the primitives that can be used to make up the rules. Three types of primitive are included, stored in three vectors. The vector *equals* stores the tests for equality that can be used. These include: equals, does not equal, greater than and less than. Then *operators* stores the mathematical operators that can be included: plus, minus, divide and multiply. Others are of course possible. Lastly *vars* stores the agent model variables that can be used. These are simply listed as "A", "B", "C" and so on. The user must decide and note what each refers to in their model. For example "A" might be agent age, "B" might be agent income, "C" might be agent relationship status. In the Hawk Dove example they refer to variables that store changeable agent behaviours and characteristics—the amount of resource that the agent took in the last round of the game, the total amount of resource that the agent currently owns and so on.

The file *RIGSS program* then implements IGSS.

Genetic programming starts with a population of random rules—the first generation of rules. Each of the rules is tested for fitness by passing it to the agent model function. Fitness is the answer to the question: if agents follow this rule, how close will the output of the agent model be to the reference dataset? A common fitness metric, and the one currently implemented, is the squared difference between agent model data and reference data. The three genetic operations are then used repeatedly to create a second generation of rules. This generation of rules is then tested for fitness. A third generation is created as the second was and so on over successive generations until a stopping condition is reached. RIGSS currently runs for a set number of generations. The top five most fit rules are stored in the vector *bestRules*.

The reference dataset usually will come from observations made on the target or it could be simulated based on some ideal situation that a researcher is interested in. For Hawk Dove a reference dataset has been created that is suitable for our illustration.

3. Impact and future developments

Inverse Generative Social Science marries the power of evolutionary computing and agent-based modelling. It has the ability to explore a vast search space of variables and their relationships seeking non-linear models to fit to data. More importantly the best models found have the potential to act as formal theory. Although rare in social science [7], it is recognised that formalising theory brings many advantages [8–15]. Implementing a theory as an agent model is one way to achieve this. By implementing a theory in this way it can be tested and explored. Running a model generates data that can be compared with observations. This will demonstrate – prove in fact – that a model, and therefore a theory, explains or does not explain those observations. (It does not prove that it is the correct explanation, only that it is an explanation.) Implementing a theory as a computational model makes an imprecise social science theory formal and precise, and allows for rigorous testing [16].

Inverse Generative Social Science reverses this and uses AI to find agent behaviour that can be considered social theory, even if it must be pruned, refined or further theorised on first. This is a very exciting prospect and one that is starting to generate interest [17–21]. However taking on an IGSS project will be an intimidating prospect for many social scientists who are not trained primarily in computer programming. RIGSS is intended as a relatively straightforward platform to begin an IGSS study and as such has good potential to make IGSS more widely practised.So far RIGSS has been used to explore the philosophy of simulation underpinning IGSS [22]. It is currently being used to study modern slavery in business supply networks, and behaviour in online child sexual exploitation and abuse. The advantage of studying targets like these with IGSS is that there may be limited theory available to create a model from because social scientists struggle to reach stakeholders as they have legitimate safety and legal liability concerns. The advantage RIGSS brings is that we can evolve rules that such stakeholders might be using, assessing them first for fit against reasonably readily available reference datasets (for instance from the Child Rescue Coalition), and second for their ability to function as theory explaining the behaviour of those involved. This has enormous potential to change these fields.

CRediT authorship contribution statement

Thomas Chesney: Writing – original draft, Software. **Robert Pasley:** Software. **Muhammad Asif Jaffer:** Software.

Declaration of competing interest

We declare no conflict of interest.

References

[1] N. Gilbert, K. Troitzsch, Simulation for the Social Scientist, second ed., Open University Press, 2005.
[2] U. Wilensky, W. Rand, An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with Netlogo, MIT Press, 2015.
[3] J.M. Epstein, I. Garibay, E. Hatna, M. Koehler, W. Rand, Special section on inverse generative social science: Guest editors' statement, J. Artif. Soc. Soc. Simul. 26 (2023) 2.
[4] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, 1992.
[5] J.R. Koza, Genetic Programming, MIT Press Cambridge, 1992.
[6] C.F. Camerer, Behavioral Game Theory: Experiments in Strategic Interaction, Princeton University Press, 2011.
[7] R. Adner, L. Polos, M. Ryall, O. Sorenson, The case for formal theory, Acad. Manag. Rev. 34 (2009) 201–208.
[8] J.R. Edwards, Reconsidering theoretical progress in organizational and management research, Organ. Res. Methods 13 (2010) 615–619.
[9] J. Davis, K. Eisenhardt, C. Bingham, Developing theory through simulation methods, Acad. Manag. J. 32 (2007) 480–499.
[10] J.B. Vancouver, X. Li, J.M. Weinhardt, P. Steel, J.D. Purl, Using a computational model to understand possible sources of skews in distributions of job performance, Personnel Psychol. 69 (2016) 931–974.
[11] J.B. Vancouver, K.B. Tamanini, R.J. Yoder, Using dynamic computational models to reconnect theory and research: Socialization by the proactive newcomer as example, J. Manag. 36 (2010) 764–793.
[12] S. Farrell, S. Lewandowsky, Computational models as aids to better reasoning in psychology, Curr. Direct. Psychol. Sci. 19 (2010) 329–335.
[13] J.R. Busemeyer, A. Diederich, Cognitive Modeling, Sage, 2010.
[14] A. Lomi, E.R. Larsen, Dynamics of Organizations: Computational Modeling and Organizational Theories, The MIT Press, 2001.
[15] J.A. Grand, M.T. Braun, G. Kuljanin, S.W. Kozlowski, G.T. Chao, The dynamics of team cognition: A process-oriented theory of knowledge emergence in teams, J. Appl. Psychol. 101 (2016) 1353.

[16] J.R. Edwards, J.W. Berry, The presence of something or the absence of nothing: Increasing theoretical precision in management research, Organ. Res. Methods 13 (2010) 668–689.

[17] T.M. Vu, C. Buckley, J.A. Duro, A. Brennan, J.M. Epstein, R.C. Purshouse, Can social norms explain long-term trends in alcohol use? Insights from inverse generative social science, J. Artif. Soc. Soc. Simul. 26 (2023) 2.

[18] L. Miranda, J. Baggio, G. Ozmen, Evolutionary model discovery of human behavioral factors driving decision-making in irrigation experiments, J. Artif. Soc. Soc. Simul. 26 (2023) 2.

[19] C. Gunaratne, E. Hatna, J.M. Epstein, I. Garibay, Generating mixed patterns of residential segregation: An evolutionary approach, J. Artif. Soc. Soc. Simul. 26 (2023) 2.

[20] R. Greig, C. Major, M. Pacholska, S. Bending, J. Arranz, Learning interpretable logic for agent-based models from domain independent primitives, J. Artif. Soc. Soc. Simul. 26 (2023) 2.

[21] J.M. Epstein, Inverse generative social science: Backward to the future, J. Artif. Soc. Soc. Simul. 26 (2023) 2.

[22] T. Chesney, A. Jaffer, R. Pasley, Examining inverse generative social science to study targets of interest, 2024, http://arxiv.org/abs/2407.13474.