An Introduction to *ABED*: Agent-Based Simulation of Evolutionary Game Dynamics. Online Appendix

Luis R. Izquierdo*, Segismundo S. Izquierdo†, and William H. Sandholm‡

October 9, 2018

Contents

I Monte Carlo simulations in *ABED* using *BehaviorSpace* **1**

II Parameter tables **4**

I. Monte Carlo simulations in *ABED* using *BehaviorSpace*

In this appendix, we explain how to run Monte Carlo experiments in *ABED* using *BehaviorSpace* (Wilensky and Shargel, 2002).[1] To create a computational experiment in *ABED*, go to *Tools* > *BehaviorSpace* in the NetLogo top menu, and click on *New* to open a window like the one shown in Figure 1. The first field in the window allows one to set a name for the experiment, while the second, larger field is where one specifies

*Department of Civil Engineering, Universidad de Burgos.
†Department of Industrial Organization, Universidad de Valladolid.
‡Department of Economics, University of Wisconsin.

[1] It is also possible to link *NetLogo* with *R* (R Core Team, 2018), *Python* (Python Software Foundation, 2018), *Mathematica* (Wolfram Research, Inc., 2018), and *Matlab* (MathWorks, Inc., 2018). Specifically, using an *R* package called *RNetLogo* (Thiele (2014); Thiele et al. (2014, 2012a,b)), one can run and control *NetLogo* models and execute *NetLogo* commands and queries from within *R*. The connector *PyNetLogo* (Jaxa-Rozen and Kwakkel, 2018) provides the same functionalty for Python, and the so-called *Mathematica link* (Bakshy and Wilensky, 2007) for Mathematica. Conversely, one can also call *R*, *Python* and *Matlab* commands from within *NetLogo* using the *R-Extension* (Thiele and Grimm, 2010), the *NetLogo Python extension* (Head, 2018) and *MatNet* (Biggs and Papin, 2013), respectively.

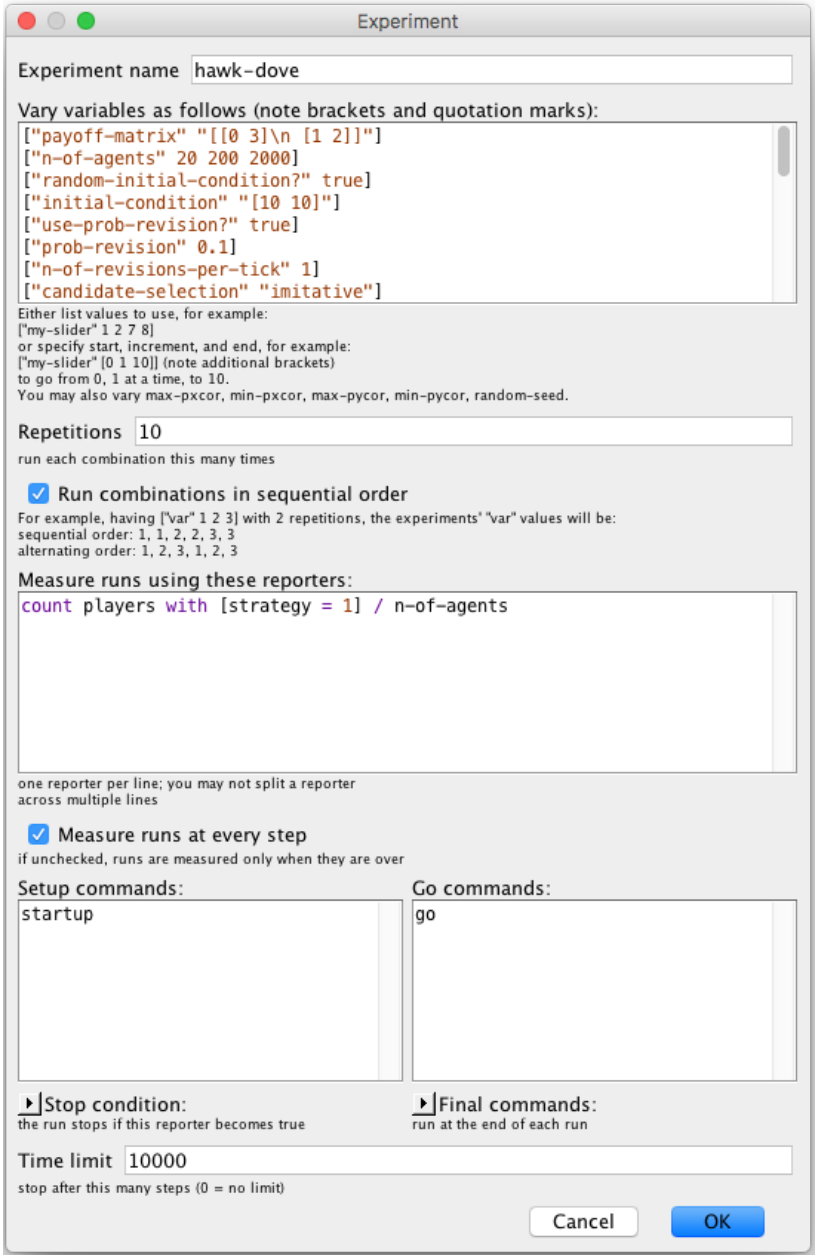

Figure 1: A *BehaviorSpace* window used to administer a computational experiment.

the parameter values to be considered. Initially, this second field contains the names of all parameters in *ABED* with their current values, surrounded by square brackets (e.g., [“`n-of-agents`” 20]). To explore several values of one parameter, one simply includes those values in the appropriate list (e.g., [“`n-of-agents`” 20 200 2000]). One can specify parameter values as a loop using the syntax [“`name-of-parameter`” [*start increment end*]]. (Note the additional brackets.) For instance, writing [“`n-of-agents`” [100 10 200]] would run simulations with 100, 110, 120, . . . , and 200 agents. If multiple values are specified for more than one parameter, then *BehaviorSpace* will consider all possible combinations of the specified parameter values. For each combination, *BehaviorSpace* will run the number of simulations specified in the “Repetitions” field.

The next field specifies the information that will be extracted from each simulation run, written in *NetLogo* language. For instance, to obtain the proportion of players using strategy 1, we should write:

```
count players with [strategy = 1] / n-of-agents
```

The information detailed in this field will be gathered either at every tick, or only at the very end of each simulation, depending on whether or not the box labeled “Measure runs at every step” is checked.

The field labeled “Setup commands” must be filled with the name of the functions that should be executed just once at the beginning of each simulation run. When using *ABED*, one should enter `startup` here. The field labeled “Go commands” includes the names of the functions that will be executed repeatedly, i.e. in every tick. In *ABED*, this field should contain `go`. One can also run specific functions at the end of each simulation run—for instance, to export plots to files—by using the field “Final commands”.

Each simulation will be run for the number of ticks specified in the field labeled “Time limit”. It is possible to define other termination conditions by writing suitable *NetLogo* code.

Once the experiment is set up, it is saved by clicking on “OK”. To run an experiment, select it in the main *BehaviorSpace* window and click on “Run”. As the experiment proceeds, *NetLogo* will save all the requested data in a *csv* file whose name and location are chosen by the user.

Finally, it is worth mentioning that computational experiments set up with *BehaviorSpace* can be run from the command line, i.e. without having to open *NetLogo*’s graphical user interface. This feature is particularly useful for launching large-scale experiments in computer clusters.

II. Parameter tables

Game, population size and initial state		
Game	**Initial state**	**Population size**
payoff-matrix = [[0 3] [1 2]]	random-initial-condition? = *off* initial-condition = [10 10]	(n-of-agents ← 20)

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 1

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *imitative* n-of-candidates = 2	complete-matching? = *off* n-of-trials = 1	decision-method = *best* tie-breaker = *stick-uniform* prob-mutation = 0

Table 1: Simulation parameters for Figure 5 (Section 3.1).

Game, population size and initial state		
Game	**Initial state**	**Population size**
payoff-matrix = [[0 −1 2] [2 0 −1] [−1 2 0]]	random-initial-condition? = *on*	n-of-agents = 50

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 1

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *imitative* (n-of-candidates ← 2)	complete-matching? = *on* (n-of-trials ← 49)	decision-method = *linear-dissatisfaction* prob-mutation = 0

Table 2: Simulation parameters for figures 6, 7, and 8 (Section 3.2).

Game, population size and initial state		
Game	**Initial state**	**Population size**
payoff-matrix = [[0 −2 1] [1 0 −2] [−2 1 0]]	random-initial-condition? = *on*	n-of-agents = 100

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 1

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *direct* n-of-candidates = 3	complete-matching? = *on* (n-of-trials ← 99)	decision-method = *best* tie-breaker = *uniform* prob-mutation = 0

Table 3: Simulation parameters for Figure 10 (Section 3.4).

Game, population size and initial state	
Game	**Initial state**
payoff-matrix = [[0 0 0 0 1] [1 0 0 0 0] [0 1 0 0 0] [0 0 1 0 0] [0 0 0 1 0]]	random-initial-condition? = *off* initial-condition = [200 200 200 200 200]
	Population size
	(n-of-agents ← 1000)

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 10

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *imitative* (n-of-candidates ← 2)	complete-matching? = *off* n-of-trials = 1	decision-method = *pairwise-difference* prob-mutation = 10^{-3}

Table 4: Simulation parameters for Figure 11 (Section 3.5). In Figure 11(ii), the parameter prob-mutation is changed to 10^{-4}.

Game, population size and initial state	
Game	**Initial state**
payoff-matrix = [[1 1 1 1] [1 1 1 1] [0 1 1 1] [0 0 1 2]]	random-initial-condition? = *off* initial-condition = [100 0 0 0] **Population size** (n-of-agents ← 100)

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 1

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *imitative* n-of-candidates = 2	complete-matching? = *on* (n-of-trials ← 99)	decision-method = *best* tie-breaker = *random-walk* prob-mutation = 10^{-3}

Table 5: Simulation parameters for figures 12 and 13 (Section 3.6). The panels of Figure 12 use four different specifications of the parameter tie-breaker.

Game, population size and initial state	
Game	**Population size**
payoff-matrix = [[[0 0] [0 0] [0 0] [0 0] [0 0] [0 0]] [[-1 3] [2 2] [2 2] [2 2] [2 2] [2 2]] [[-1 3] [1 5] [4 4] [4 4] [4 4] [4 4]] [[-1 3] [1 5] [3 7] [6 6] [6 6] [6 6]] [[-1 3] [1 5] [3 7] [5 9] [8 8] [8 8]] [[-1 3] [1 5] [3 7] [5 9] [7 11] [10 10]]]	pop-1-n-of-agents = 500 pop-2-n-of-agents = 500 **Initial state** random-initial-condition? = *on*

Assignment of revision opportunities	
use-prob-revision? = *off*	n-of-revisions-per-tick = 50

Revision protocol		
Candidate selection	**Matching**	**Decision method**
candidate-selection = *direct* pop-1-n-of-candidates = 6 pop-2-n-of-candidates = 6	complete-matching? = *off* n-of-trials = 1 single-sample? = *on*	decision-method = *best* tie-breaker = *min* prob-mutation = 10^{-3}

Table 6: Simulation parameters for Figure 15 (Section 3.8). The parameter single-sample? is switched from *on* to *off* mid-run.

References

Bakshy, E. and Wilensky, U. (2007). NetLogo-Mathematica link. Software. `http://ccl.northwestern.edu/netlogo/mathematica.html`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Biggs, M. B. and Papin, J. A. (2013). Novel multiscale modeling tool applied to pseudomonas aeruginosa biofilm formation. *PLOS ONE*, 8(10).

Head, B. (2018). Netlogo python extension. Software. https://github.com/qiemem/PythonExtension.

Jaxa-Rozen, M. and Kwakkel, J. H. (2018). PyNetlogo: Linking Netlogo with Python. *Journal of Artificial Societies and Social Simulation*, 21(2):4.

MathWorks, Inc. (2018). Matlab. Software. https://mathworks.com. Natick, Massachusetts.

Python Software Foundation (2018). Python. Software. http://www.python.org.

R Core Team (2018). R: A language and environment for statistical computing. Software. https://www.R-project.org.

Thiele, J. C. (2014). R marries NetLogo: Introduction to the RNetLogo package. *Journal of Statistical Software*, 58(2):1–41.

Thiele, J. C. and Grimm, V. (2010). NetLogo meets R: Linking agent-based models with a toolbox for their analysis. *Environmental Modelling & Software*, 25(8):972 – 974.

Thiele, J. C., Kurth, W., and Grimm, V. (2012a). Agent-based modelling: Tools for linking NetLogo and R. *Journal of Artificial Societies and Social Simulation*, 15(3):8.

Thiele, J. C., Kurth, W., and Grimm, V. (2012b). RNetLogo: An R package for running and exploring individual-based models implemented in NetLogo. *Methods in Ecology and Evolution*, 3(3):480–483.

Thiele, J. C., Kurth, W., and Grimm, V. (2014). Facilitating parameter estimation and sensitivity analysis of agent-based models: A cookbook using NetLogo and R. *Journal of Artificial Societies and Social Simulation*, 17(3):11.

Wilensky, U. and Shargel, B. (2002). BehaviorSpace. Software. `http://ccl.northwestern.edu/netlogo/behaviorspace.html`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wolfram Research, Inc. (2018). Mathematica. Software. https://www.wolfram.com. Champaign, Illinois.