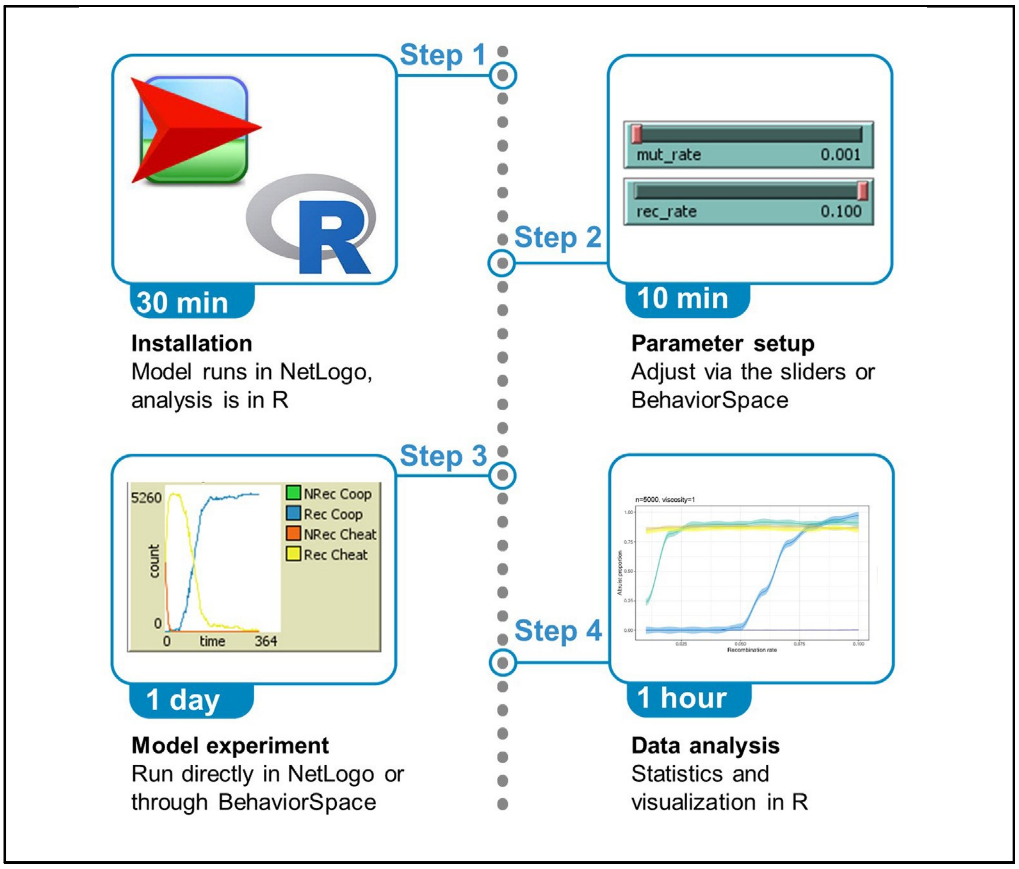
Protocol

Protocol for an agent-based model of recombination in bacteria playing a public goods game



Agent-based models are composed of individual agents coded for traits (e.g., cooperation and cheating) that interact in a virtual world based on defined rules. Here, we describe the use of an agent-based model of homologous recombination in bacteria playing a public goods game. The model is defined by the following variables: population size, population viscosity, cooperation multiplier, recombination rate, and fitness cost. This protocol is useful in analyses of horizontal gene transfer, bacterial sociobiology, and game theory.

Isaiah Paolo A. Lee, Omar Tonsi Eldakar, J. Peter Gogarten, Cheryl P. Andam

ialee1@up.edu.ph (I.P.A.L.)
candam@albany.edu (C.P.A.)

Highlights

An agent-based model is used to study cooperative behavior in bacterial populations

Modeling of homologous recombination in bacteria playing a public goods game

Implement the model in NetLogo and summarize the results using R language

Values of different population parameters and number of generations can be adjusted

Lee et al., STAR Protocols 4, 102733
December 15, 2023 ª 2023 The Author(s).
https://doi.org/10.1016/j.xpro.2023.102733

CellPress
OPEN ACCESS

Protocol

Protocol for an agent-based model of recombination in bacteria playing a public goods game

Isaiah Paolo A. Lee,[1,2,6,*] Omar Tonsi Eldakar,[3] J. Peter Gogarten,[4] and Cheryl P. Andam[5,7,*]

[1]Department of Molecular, Cellular and Biomedical Sciences, University of New Hampshire, Durham, NH 03824, USA

[2]National Institute of Molecular Biology and Biotechnology, University of the Philippines, Diliman 1101, Philippines

[3]Department of Biological Sciences, Nova Southeastern University, Fort Lauderdale, FL 33328, USA

[4]Department of Molecular and Cell Biology, University of Connecticut, Storrs, CT 06269, USA

[5]Department of Biological Sciences, University at Albany, State University of New York, Albany, NY 12222, USA

[6]Technical contact: ialee1@up.edu.ph

[7]Lead contact

*Correspondence: ialee1@up.edu.ph (I.P.A.L.), candam@albany.edu (C.P.A.)
https://doi.org/10.1016/j.xpro.2023.102733

SUMMARY

Agent-based models are composed of individual agents coded for traits, such as cooperation and cheating, that interact in a virtual world based on defined rules. Here, we describe the use of an agent-based model of homologous recombination in bacteria playing a public goods game. We describe steps for software installation, setting model parameters, running and testing models, and visualization and statistical analysis. This protocol is useful in analyses of horizontal gene transfer, bacterial sociobiology, and game theory.
For complete details on the use and execution of this protocol, please refer to Lee et al.[1]

BEFORE YOU BEGIN

Timing: <30 min

This section includes the minimal hardware requirements, installation procedures, and the format of the input files to be processed by NetLogo and R.

Hardware

We ran the following model on a computer with 16.0 GB of local memory and a 2.20 GHz processor. The model can be run on a computer with lower specifications if the model population size remains low (n < 10,000).

Software

1. Download and install NetLogo from https://ccl.northwestern.edu/netlogo/download.shtml on your computer.
   a. While the code in this experiment was run in NetLogo v.6.1.1, it will work with more recent versions of NetLogo using their 2D platforms.
   b. Choose the download appropriate for your operating system.

   Note: We ran the code on a 64-bit Windows system, Mac OS X system, and the NetLogo Web interface.

2. Launch the model.
   a. Download the file Coop_model.nlogo from https://zenodo.org/record/8431196.
   b. Open the file with NetLogo.

   Note: This can be done by clicking the downloaded file or by first opening the installed NetLogo program, such as NetLogo v.6.3.0, then clicking File -> Open. and navigating to Coop_model.nlogo.

   Optional: The output files from the BehaviorSpace tool in NetLogo are in tabular format (.csv) and can be loaded into R. For further analysis, download and install the R programming language from https://cran.r-project.org/mirrors.html on your computer.

While the statistical analysis in this experiment was run in R v.3.6.3, it will also work with more recent versions of R.

For visualization of data, install the packages ggplot2, pals, and patchwork.

While we used older version of these packages, the required functions are still available in more recent versions. This can be done by running the code below.

```
>install.packages("ggplot2")

>install.packages("pals")

>install.packages("patchwork")
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Code and data for recombination modeling	Lee[2]	https://datadryad.org/stash/dataset/doi:10.5061/dryad.9ghx3ffnc
Software and algorithms		
NetLogo v.6.1.1	Wilensky[3]	http://ccl.northwestern.edu/netlogo/
ggplot2 v.3.3.3	Wickham[4]	https://ggplot2.tidyverse.org
pals v.1.6	Wright[5]	https://kwstat.github.io/pals/
patchwork v.1.1.1	Pedersen[6]	https://patchwork.data-imaginist.com https://github.com/thomasp85/patchwork
R language v.3.6.3	R Core Team[7]	https://www.R-project.org/

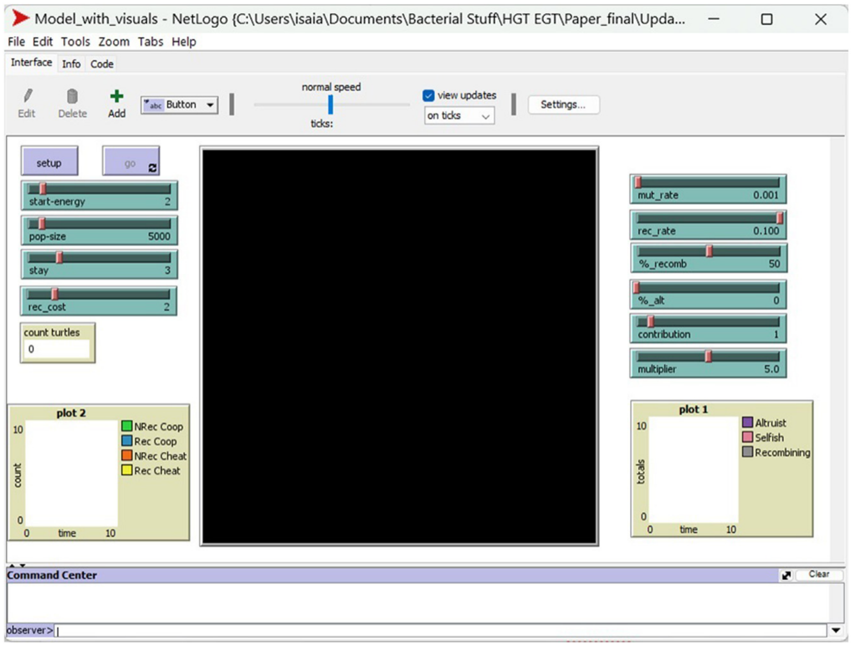
STEP-BY-STEP METHOD DETAILS

Setting model parameters

Timing: 10 min

This step sets the range of model parameters (population size, population viscosity, recombination cost, mutation rate, recombination rate, proportion of the population consisting of recombining cells, proportion of the population consisting of cooperating cells) to be used for subsequent analysis.

1. Adjust the model parameters using the sliders.



start-energy 2
pop-size 5000
stay 3
rec_cost 2
count turtles 0
mut_rate 0.001
rec_rate 0.100
%_recomb 50
%_alt 0
contribution 1
multiplier 5.0

Figure 1. NetLogo model interface
The code, when loaded into NetLogo appears as follows. Model parameters can be adjusted using the sliders. The central black area is where the cells will appear when the model is initialized.

Note: The model comes with sliders to select each parameter value from a range of values (Figure 1). The sliders themselves can be optionally modified with the following steps.

a. Right-click a slider.
b. Select Edit.
c. Specify new minimum, maximum, and increment values.

2. Adjust the model space.

Note: The default model space is a 32 unit by 32 unit two-dimensional torus.

a. Right-click the central display area.
b. Select Edit.
c. The size of the world can be changed by editing the max-pxcor and max-pycor values.
d. Horizontal and vertical wrapping can be changed using the tick boxes.
e. Other visualization settings can also be adjusted.

3. Set up the population parameters using the sliders.
a. Select the starting resource values of each cell using the start-energy slider.

Note: Energy is consumed over time. When this reaches zero, a cell dies. The relative reproduction rates of each type of cell also depend on these values. We used a default initial value of two.

b. Select the population size using the pop-size slider.

Note: When cells die, remaining cells reproduce to keep the population size constant across generations. We set this at 3000, 5000, and 7000 for our study since the model slows down as the population size increases. Reproducing cells are chosen randomly, with the probability of reproduction proportional to the relative fitness.

c. Select the population viscosity value, i.e., the number of generations it takes for an individual cell to disperse, using the stay slider.

Note: Viscosity is the number of generations each cell will remain in the same location before moving to an adjacent location. We set this at 1, 2, and 3 for our study.

d. Select the constitutive cost of being a recombiner using the rec_cost slider.

Note: Constitutive cost subtracts from the resources available to each cell every generation. At 0, recombining and non-recombining cells have the same energy expenditure. We set this at 0, 1, and 2 for our study.

e. Select the mutation rate using the mut_rate slider.

Note: Mutation rate is the rate at which cooperative cells produce cheater offspring and vice versa. At 0, replication is always faithful. We set this at 0.001 for our study.

f. Select the recombination rate using the rec_rate slider.

Note: Recombination cost is the rate at which recombining cells change the phenotype of other cells in the same location. Cooperative cells make other cells cooperate, and cheater cells make other cells cheat. Converted cells also become recombiners themselves.

Note: At 0, recombining cells and non-recombining cells exhibit the same behavior. We set this from 0.01 to 0.10 for our study.

g. Select the initial percentage of recombiners using the %_recomb slider.

Note: We set this at 50 for our study, indicating equal starting percentages of recombining and non-recombining cells.

h. Set the initial percentage of cooperating cells using the %_alt slider.

Note: We set this at 0 for our study to model the invasion of cooperation in an established cheating population.

i. Select the fitness benefit to the entire group.

Note: At each generation, all cells in each location engage in a public goods game.[8]

   i. Select the initial fitness value using the contribution slider.

Note: At the start, all cooperators contribute some amount of fitness to a common pool. We set this value at 1 for our study.

   ii. Select the benefit of cooperation using the multiplier slider.

Note: The common pool value is amplified, signifying the benefit of cooperation. The multiplier slider sets the number that the common pool value is multiplied with. At 1, there is no benefit to cooperation. We set this from 1 to 5 for our study.

Running a single model in interactive mode

Timing: 5 min

The model will run with the parameters set above. The progress of the population through each generation can be monitored in real time through the interface.

4. Click the setup button. This initializes the population according to the parameters selected above.
   a. The model speed can be adjusted with the bar above the display.
5. Click the go button to start the model. To pause the model, click the go button again.
6. View the population on the middle display, with each cell represented by a triangle.
   a. Green triangles are non-recombining cooperators.
   b. Blue triangles are recombining cooperators.
   c. Orange triangles are non-recombining cheaters.
   d. Yellow triangles are recombining cheaters.

   Note: A video showing the display can be viewed at https://www.cell.com/cms/10.1016/j.isci.2023.107344/attachment/2756d2e1-82e5-4af7-ad57-79e1aa5413ee/mmc3.mp4

7. View the number of each type of cell at each time point on the graph on the left. See Figure 2 for example.
   a. The green line shows the number of non-recombining cooperators.
   b. The blue line shows the number of recombining cooperators.
   c. The orange line shows the number of non-recombining cheaters.
   d. The yellow line shows the number of recombining cheaters.
8. Stop the model.

   Note: The model will run until stopped. For most parameter sets we tested, the population compositions stabilized after 500 generations.

Using BehaviorSpace

Timing: >1 h

This step uses the built-in BehaviorSpace tool in NetLogo to test a model multiple times (replicates).

9. Click Tools > BehaviorSpace > New to be able to set up a parameter sweep experiment.
10. Define the variable ranges to test.

    Note: The parameters that can be changed in the model are the same as those defined in Steps 2 and 3.

    a. Select the values by listing them in brackets, as shown in the example below.

```
["pop-size" 3000 5000 7000]

["%_alt" 0]

["rec_cost" 0]

["%_recomb" 50]

["mut_rate" 0.001]
```

```
["start-energy" 2]

["stay" 1]

["contribution" 1]
```

b. Select the ranges to be tested in brackets within brackets using the start, increments, and end, as shown in the example below.

```
["multiplier" [1 1 5]] ["rec_rate" [0.01 0.01 0.1]]
```

c. Specify the number of repetitions.

Note: In our study, we used 100 repetitions.

d. Specify the populations of each cell type by listing them as reporters for the runs.

Note: For example, the following measures count the number of cooperators, cheaters, re-combining cooperators, and recombining cheaters.

```
count As

count Ss

count As with [recomb = 1]

count Ss with [recomb = 1]
```

e. Click the box for "Measure runs at every step box" to record the population each generation.

Note: By default, the populations are recorded at the end of each run.

f. Specify the time limit, i.e., the number of generations the model runs for.

Note: We set this at 500 for our study.

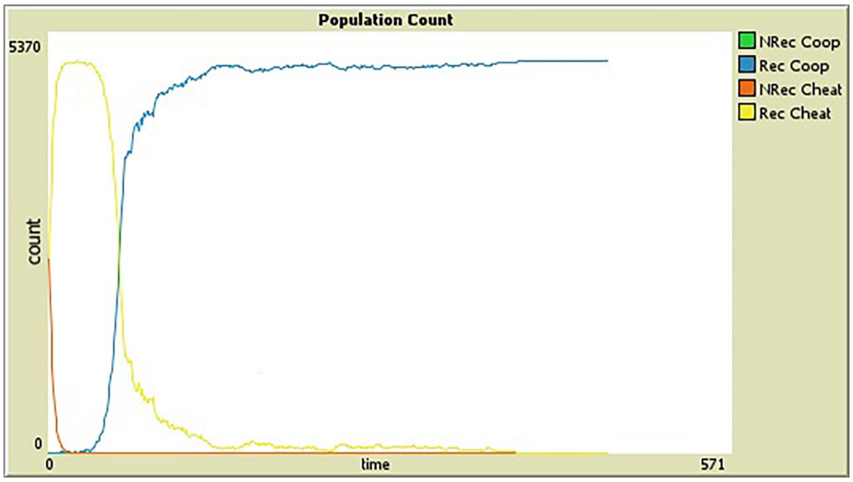


Figure 2. Sample population graph

g. Once the parameters have been defined, click ''Ok and Run'' to run it.

Note: The output can be saved in different formats, but a table output in csv format is what we ran for further statistical analysis in R.

h. View the output table.

Note: The table output has six header lines describing the batch run, followed by the column names of the data rows, followed by the data.

Note: For the downstream analysis in the code below, we removed the first six lines of this file. We then removed the columns [run number], %_alt, %_recomb, mut_rate, start-energy, contribution, and step.

Visualization and statistical analysis

Timing: 1 h

This step generates a visualization of the runs from each set of parameters replicated 100 times.

11. Generate local polynomial regression curves using the stat_smooth function of ggplot2, with the parameters method = "loess" and span = 0.2.

Note: The curves take into account 20% of the total data closest to each point for fitting each location. The bands around the curves indicate the 95% confidence interval, also from the stat_smooth function.

12. View the runs from each set of parameters and 100 replicates.

Note: Sample graphs and the code to generate them given an input file from BehaviorSpace named Compiled_runs.csv are shown in Figures 3 and 4.

Note: Download the processed BehaviorSpace output file we used from Dryad at https://datadryad.org/stash/dataset/doi:10.5061/dryad.9ghx3ffnc.

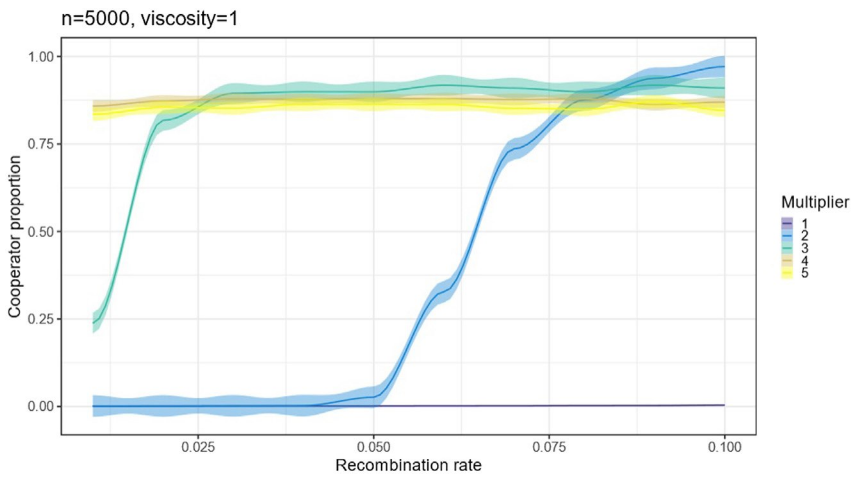


Figure 3. Sample plot of BehaviorSpace output for different values of cooperation multiplier (range: 1 - 5)
The bands show the 95% confidence interval. Figure adapted with permission from Lee et al., 2023.[1]

```
mesa=read.csv("Compiled_runs.csv") # From BehaviorSpace output

colnames(mesa) = c("pop_size","rec_cost","mult","stay",

            "rec_rate","As","Ss","RAs","RSs")

mesa$prop_A = mesa$As /  mesa$pop_size

# For a single plot

library(ggplot2)

library(pals)

# Function to draw a ribbon plot given a population size,

# recombination cost, and population viscosity

draw.recr <- function (poop, coost, ste) {

  grape=ggplot(data  =  mesa[which(mesa$pop_size==poop  &  mesa$rec_cost==coost  &  mesa$
stay==ste),],

          aes(x =rec_rate, y=prop_A, group=mult,

            color =as.factor(mult), fill =as.factor(mult))) +

    scale_color_manual(name="Multiplier",values=parula(5)) +

    scale_fill_manual(name="Multiplier",values=parula(5)) +

    stat_smooth(se=T, method="loess", span=0.2, geom="smooth",n=100,level=0.95) +

  #geom_smooth(se=T, method="loess", span=0.2) +

  #geom_point() +

  theme_bw(base_size=25) +

  labs(y="Cooperator proportion", x ="Recombination rate")

return(grape)

}

draw.recr(poop=5000,coost=0,ste=1)+ggtitle("n=5000,
viscosity=1")+theme_bw(base_size=25)

# For multiple plots

library(patchwork)

ploot1=draw.recr(poop=3000,coost=0,ste=1)+ggtitle("n=3000,
viscosity=1")+theme_bw(base_size=25)

ploot2=draw.recr(poop=5000,coost=0,ste=1)+ggtitle("n=5000,
viscosity=1")+theme_bw(base_size=25)

ploot3=draw.recr(poop=7000,coost=0,ste=1)+ggtitle("n=7000,
viscosity=1")+theme_bw(base_size=25)

ploot4=draw.recr(poop=5000,coost=0,ste=2)+ggtitle("n=5000,
viscosity=2")+theme_bw(base_size=25)

ploot5=draw.recr(poop=5000,coost=0,ste=3)+ggtitle("n=5000,
viscosity=3")+theme_bw(base_size=25)

ploot1+ploot2+ploot3+ploot4+ploot5+guide_area() +

  plot_layout(guides ='collect') +

  plot_annotation(tag_levels="A")
```

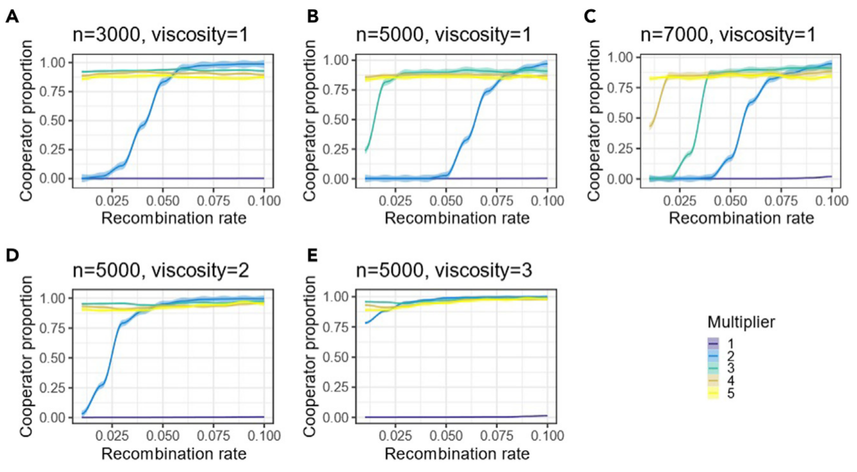


Figure 4. Sample plot of BehaviorSpace output for different values of cooperation multiplier under different parameter sets

The bands show the 95% confidence interval. Figure adapted with permission from Lee et al., 2023.[1]

EXPECTED OUTCOMES

The expected output from the prior R code is a ribbon plot or a series of ribbon plots depicting how the proportion of altruists in the population changes with recombination rate. Other variables can also be monitored and plotted in a similar way. See Figures 3 and 4 for examples of BehaviorSpace output for different sets of parameters.

LIMITATIONS

Due to this being a computational model, some assumptions are made about bacterial populations that may not hold in real life. For example, bacterial recombination rates may vary within the same population. The benefits of cooperation may also be context dependent as opposed to the example in the model represented by a fixed multiplier. Hence, caution must be exercised when inferring results from this model. Agent-based models also operate more slowly than other types of models; hence, increasing the population size may slow down the total time to run the model.

TROUBLESHOOTING

Problem 1

BehaviorSpace variables tested are not those intended (steps 10.a and 10.b).

Potential solution

Check that the variable names are correctly spelled and enclosed in quotation marks. Make sure that variable ranges are specified in another set of square brackets, so they do not get interpreted as lists.

Problem 2

BehaviorSpace runs too slowly (step 10.g).

Potential solution

Do not update the graphics. Only record populations at the end of each run, not every step. If applicable, lower the population size and area.

Problem 3

The output graphs are too jagged and have wide confidence intervals (step 11).

Potential solution

Increase the number of BehaviorSpace replicates. The parameters of the curve smoothing function in R can also be adjusted (such as span). If the runs do not appear to stabilize, increase the number of generations for the experiment.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to the lead contact, Cheryl P. Andam (candam@albany.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- All original code and data can be accessed in Dryad and are publicly available: https://datadryad.org/stash/dataset/doi:10.5061/dryad.9ghx3ffnc
- Instructions for running the simulations are also provided at the same Dryad site.
- Additional information required to reanalyze the data reported in this paper is available from the lead contact upon reasonable request.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (award no. 2055120) to C.P.A. J.P.G. is supported by the National Science Foundation (award no. 1716046). I.P.A.L. is funded by the UP System Balik PhD Program (OVPAA-BPhD-2022-03) and was also supported by the University of New Hampshire 2021 Summer Teaching Assistant Fellowship. We also thank Joshua Skydel (University of Connecticut) for help with setting up the NetLogo model.

AUTHOR CONTRIBUTIONS

Conceptualization, C.P.A. and I.P.A.L.; methodology, I.P.A.L.; formal analysis, I.P.A.L.; writing – original draft, C.P.A. and I.P.A.L.; writing – review and editing, C.P.A., I.P.A.L., O.T.E., and J.P.G.; supervision, C.P.A.; funding acquisition, C.P.A., I.P.A.L., and J.P.G.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Lee, I.P.A., Eldakar, O.T., Gogarten, J.P., and Andam, C.P. (2023). Recombination as an enforcement mechanism of prosocial behavior in cooperating bacteria. iScience 26, 107344.
2. Lee, I.P.A., Eldakar, O.T., Gogarten, J.P., and Andam, C. (2023). Data from: Recombination as an Enforcement Mechanism of Prosocial Behavior in Cooperating Bacteria. 1720006 bytes.
3. Wilensky, U. (1999). NetLogo.
4. Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis (Springer-Verlag).
5. Wright, K. (2021). Pals: Color Palettes, Colormaps, and Tools to Evaluate Them.
6. Pedersen, T.L. (2020). Patchwork: The Composer of Plots.
7. R Core Team (2021). R: The R Project for Statistical Computing.
8. Fehr, E., and Gachter, S. (2000). Cooperation and punishment in public goods experiments. Am. Econ. Rev. 90, 980–994.