Chapter 16 Exercise Hints and Solutions

Agent-based and Individual-Based Modeling: *A Practical Introduction, 2nd Edition*

Exercise 1

An Excel workbook implementing the logistic function is provided: `Ch16-Ex1_LogisticFunc_2ndEd.xlsx`.

Exercise 2

The information in Section 16.4.1 is sufficient to write a complete ODD description; however, some interpretation is required to describe the design concepts accurately. (We do not provide a completed ODD.)

Exercise 3

The implementation `WildDogs_Ch16-Ex3_2ndEd.nlogo` is provided.

This model is particularly challenging to program and test because of the multiple kinds of entities (packs, disperser groups, dogs). Common mistakes that can be hard to find include:

- Misspelling dog status (e.g., `alfa` in some places and `alpha` in others);
- Neglecting to link packs or disperser groups to the dogs belonging to them when packs reproduce and when packs or disperser groups are formed; and
- When creating a pack from disperser groups, having one disperser group die before asking the other group to die (execution of code stops the instant the agent executing it dies).

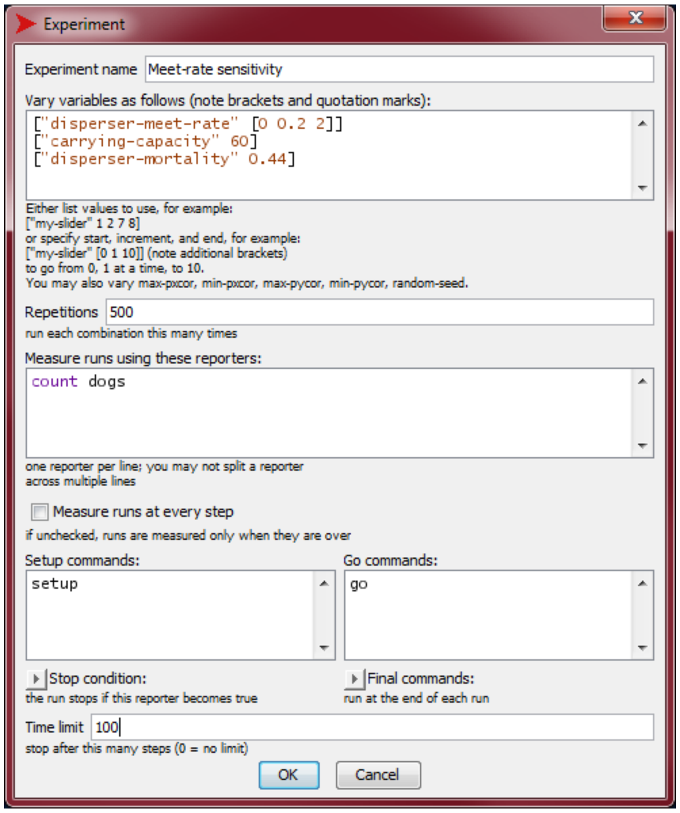
Code for models such as this should be tested by using a variety of techniques throughout the code, that check individual statements or procedures or just look for problems. The program we provide includes:

- Simple test outputs used temporarily as the program is written.
- Defensive programming checks throughout. For example: do packs have no more than 2 alpha individuals? Does any dog have value of `status` other than the legal ones? Are there any dogs not linked to a pack or disperser group, or any collectives not linked to dogs?
- A procedure `test-packs` that writes to an output file the status of each pack: how many members it has of each social status, and the actual status variables of each member. Output to this file can be produced between the major actions: age and status updates, selecting alphas, reproduction, dispersal, etc. The file can then be examined for events that violate model rules.

There has been discussion about whether the model description is ambiguous concerning whether members of a newly formed pack could leave it again in a disperser group, during the same time step. The model description does not address this possibility explicitly, but the answer is in the model's schedule. Pack formation occurs *after* dispersal, so packs should not be subject to dispersal in the year they were formed.

Exercise 4

The frequency of extinction within 100 years can be examined with BehaviorSpace experiments like this:



Experiment

Experiment name: Meet-rate sensitivity

Vary variables as follows (note brackets and quotation marks):

```
["disperser-meet-rate" [0 0.2 2]]
["carrying-capacity" 60]
["disperser-mortality" 0.44]
```

Either list values to use, for example:
["my-slider" 1 2 7 8]
or specify start, increment, and end, for example:
["my-slider" [0 1 10]] (note additional brackets)
to go from 0, 1 at a time, to 10.
You may also vary max-pxcor, min-pxcor, max-pycor, min-pycor, random-seed.

Repetitions: 500
run each combination this many times

Measure runs using these reporters:

```
count dogs
```

one reporter per line; you may not split a reporter across multiple lines

Measure runs at every step [ ]
if unchecked, runs are measured only when they are over

Setup commands:

```
setup
```

Go commands:

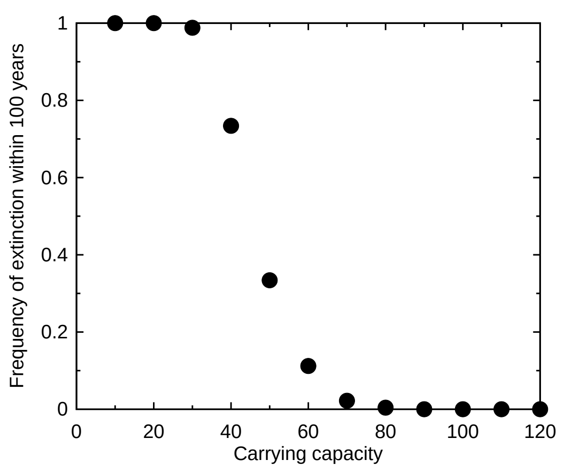
```
go
```

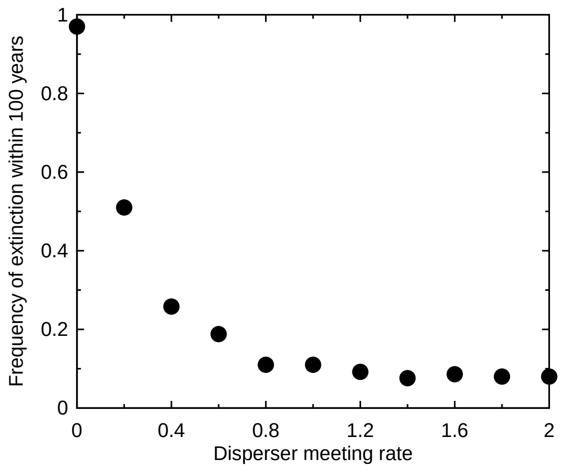
Stop condition:
the run stops if this reporter becomes true

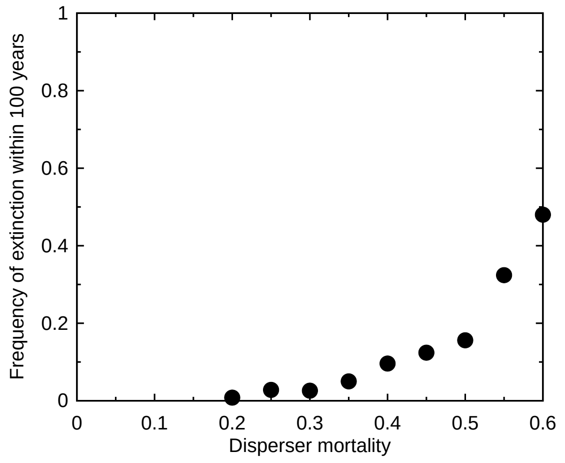
Final commands:
run at the end of each run

Time limit: 100
stop after this many steps (0 = no limit)

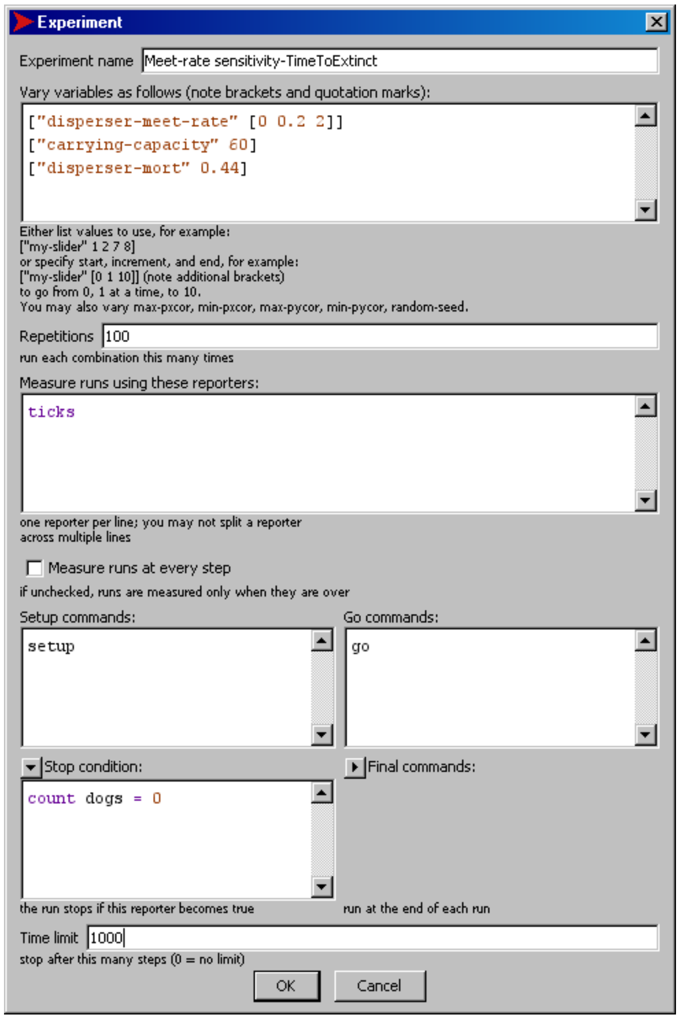
The model is run for 500 replicates of each parameter value, writing out only the number of dogs alive at the end of 100 years. (The "Time limit" of 100 in BehaviorSpace is not necessary if there is a stopping rule in the go procedure.) Then, in analysis software (e.g., a spreadsheet), a new variable can be created and set to 1.0 if the dogs went extinct (`count dogs` = 0) or to 0 if not; averaging that variable over the 500 repetitions gives a frequency of extinction. Excel's PivotTables are useful for this kind of analysis. Results should look like the following graphs. (The experiment requires changing the mortality probability for dispersers from a constant 0.44 to a global variable on an interface slider.)


Frequency of extinction within 100 years
Carrying capacity


Frequency of extinction within 100 years
Disperser meeting rate


Frequency of extinction within 100 years
Disperser mortality

The mean time to extinction can be evaluated using an experiment setup like this:



Experiment name: Meet-rate sensitivity-TimeToExtinct

Vary variables as follows (note brackets and quotation marks):

```
["disperser-meet-rate" [0 0.2 2]]
["carrying-capacity" 60]
["disperser-mort" 0.44]
```

Either list values to use, for example:
["my-slider" 1 2 7 8]
or specify start, increment, and end, for example:
["my-slider" [0 1 10]] (note additional brackets)
to go from 0, 1 at a time, to 10.
You may also vary max-pxcor, min-pxcor, max-pycor, min-pycor, random-seed.

Repetitions: 100
run each combination this many times

Measure runs using these reporters:

```
ticks
```

one reporter per line; you may not split a reporter across multiple lines

Measure runs at every step: ☐
if unchecked, runs are measured only when they are over

Setup commands:

```
setup
```

Go commands:

```
go
```

Stop condition:

```
count dogs = 0
```

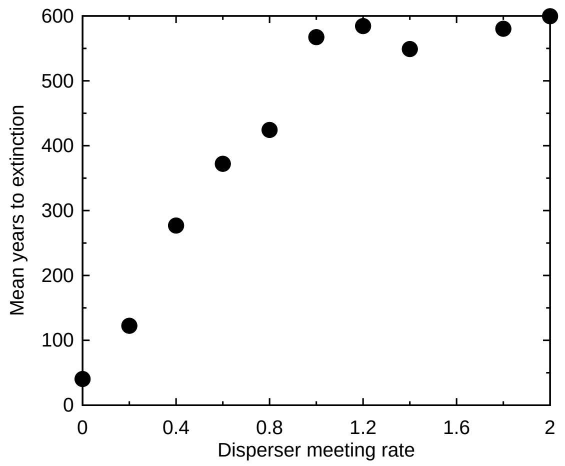
the run stops if this reporter becomes true

Final commands:

run at the end of each run

Time limit: 1000
stop after this many steps (0 = no limit)

The stopping rule in the code (stop at 100 years) must be removed because the new rule (stop when the population is extinct; or at 1000 years) is now in the Experiment Setup. The year at which the model stops is therefore the time at extinction. Example results from this experiment are:



(The shape of this curve might be different if the upper limit on simulation time was greater than 1000 years.)

The disperser meeting rate, carrying capacity, and disperser mortality all strongly affect the persistence of the simulated dog population. However, students should point out the nonlinearities in these responses: meeting rate, for example, has much less effect after it reaches about 1.0. Extinction becomes rare when carrying capacity is above 80.

Exercise 5

A version of the Wild Dog model with transients is provided as `WildDogs_Ch16-Ex5_Transients_2ndEd.nlogo`. Some things to watch for in programming this change:

- Dogs can be transformed into transients (and vice versa) via the statement `set breed transients`. But remember, at the same time, to also re-set any dog/transient variables that change at the same time, including social status and display variables.
- When transients are created, they need to make their disperser group “die” because all dispersers turn into transients.
- Transient mortality cannot just be added to the mortality procedure for dogs because that procedure is called by a statement in the `go` procedure such as: `ask dogs [do mortality]`. Transients are no longer dogs, so that statement does not cause them to execute `do-mortality`.
- Transients need to be added to the population graph; just use `plot count transients`.
- The code we provide shows how to use file output to test the addition of transients to a pack by checking the state of a pack before, then after, a transient joins it.

In repeating the experiments analyzing sensitivity of frequency of extinction to carrying capacity and disperser meeting rate, we found that extinction occurred much less frequently once transients were added. (Varying each of these parameters by itself, as in top two graphs provided above for Exercise 4, we found no extinction at all when transients are included.)

(Here is an alternative to this exercise. In the original model, if a pack is missing an alpha and lacking subordinates of the right sex to replace that alpha, it stops reproducing and soon dies. Modify the model so such packs can obtain an alpha by combining with a disperser group. Does this change improve the population's persistence as much as adding transients does?)

Exercise 6

The point of this exercise is to make students think about how representing behaviors as belonging to individuals, instead of being behaviors of collectives, can make a model more complex. For example, it would be tempting to assume that the likelihood of a particular subordinate dog becoming an alpha (its viability in competing to become alpha) depends on its characteristics such as age or size; that would require adding rules to define exactly how the probability of becoming alpha depends on these variables. And: if size is used, then the model must be modified to represent the growth of individuals, which would require a major increase in complexity unless done in an extremely simple—and meaningless—way.

Students are likely to discover that even simple assumptions about the viability of each subordinate are not trivial to program into a decision of which becomes alpha, without some kind of pack-level statement such as "pick the pack member with highest viability".