The Economics of Egg Trading: Mating Rate, Sperm Competition and Positive Frequency-Dependence

Jonathan M. Henshaw, Michael D. Jennions, Hanna Kokko

SUPPLEMENTARY MATERIAL

Here we present the computer code for the simulation model. It is written for Wolfram Mathematica v.8 and was used to generate data for Figures 1 and 2 in the main text. Comments are given in **bold**, while the code itself appears in regular typeface.

- - - - -

m = 2; **(* Mating rate *)**

initT = 0.1; **(* Initial proportion of traders *)**

tau = 0.0; **(* Window of time for streakers to join a mating pair *)**

d = 0.5; **(* Amount by which a streaker's paternity is discounted relative to the male-role mate *)**

initPopSize = 500; **(* Initial size of the population *)**

k = 500; **(* Approximate carrying capacity of the population *)**

(* Initialise the population. Each individual is represented by a vector {a,b,c}, where:
a = 1 for traders, a = 0 for non-traders;
b is the number of eggs the individual is carrying (initially zero);
c is the time of last mating (initially zero) *)

pop = Array[{Boole[#/initPopSize <= initT], 0, 0} &, initPopSize];

time = 0; **(* Time starts at zero *)**

event = 0; **(* The number of events starts at zero *)**

fixedQ = 2; **(* 'fixedQ' determines whether trading has gone to fixation (value 1), gone extinct (value 0) or neither (value 2) *)**

(* The loop below generates a series of matings and deaths until either trading or non-trading goes to fixation *)

While[fixedQ == 2,

PopSize = Length[pop]; **(* Count the number of individuals in the population *)**

pop = Permute[pop, RandomPermutation[PopSize]]; **(* Randomise the order of individuals in the population *)**

(* Generate random variables for the times until the next death and the next mating. The smaller of these two values determines which event occurs first *)

NextDeath = RandomReal[ExponentialDistribution[PopSize^2/k]];
NextMating = RandomReal[ExponentialDistribution[m*PopSize^2/(2k)]];

time = time + Min[NextDeath, NextMating]; **(* Increase the current time by the time elapsed since the last event *)**

If[NextDeath < NextMating, **(* If death occurs first... *)**

pop = pop[[1 ;; -2]], **(* ...then remove the last individual from the population*)**

(* Otherwise, mating occurs first, so generate the number of streakers as a Poisson random variable *)

S = If[tau == 0, 0, Min[RandomInteger[PoissonDistribution[tau*m*PopSize/k]], PopSize − 2]];

maters = pop[[1 ;; 2 + S]]; **(* Select two mating individuals plus streakers from the population *)**

(* Increase the number of eggs of each individual in the mating pair by a Poisson random variable with mean equal to the time since last mating *)

maters[[1 ;; 2, 2]] = maters[[1 ;; 2, 2]]
+ Map[RandomInteger[PoissonDistribution[#]] &, (time − maters[[1 ;; 2, 3]])];

maters[[1 ;; 2, 3]] = time; **(* Set the mating individuals' time of last mating to the current time *)**

(* Calculate the number of eggs offered by each mate *)

(* If the first individual is a trader, then it offers zero eggs if its partner is not carrying any eggs. If the partner is carrying eggs, the trader gives the minimum of its own egg number and its partner's egg number plus one. If the first individual is a non-trader, it gives all of its eggs. *)

eggs1 = If[maters[[1, 1]] == 1, If[maters[[2, 2]] == 0, 0,
Min[maters[[1, 2]], maters[[2, 2]] + 1]], maters[[1, 2]]];

(* Similarly, if the second individual is a trader, then it offers zero eggs if its partner is not carrying any eggs. If the partner is carrying eggs, the trader gives the minimum of its own egg number and its partner's egg number. If the second individual is a non-trader, it gives all of its eggs. *)

eggs2 = If[maters[[2, 1]] == 1, If[maters[[1, 2]] == 0, 0,
Min[maters[[1 ;; 2, 2]]]], maters[[2, 2]]];

maters[[1 ;; 2, 2]] = maters[[1 ;; 2, 2]] − {eggs1, eggs2}; **(* Reduce each mate's egg total by the number offered *)**

pop[[1 ;; 2]] = maters[[1 ;; 2]]; **(* Update the population vector *)**

(* Assign paternity for each set of eggs based on a weighted raffle, with sneakers' share of paternity discounted by d *)

```
weights = Join[{1}, Array[d &, S]];
suitors1 = Join[{2}, Array[2 + # &, S]];
suitors2 = Join[{1}, Array[2 + # &, S]];

paternity1 = Array[RandomChoice[weights -> suitors1] &, eggs1];
paternity2 = Array[RandomChoice[weights -> suitors2] &, eggs2];
```

(* Generate the offspring resulting from each set of eggs. The gene for trading behaviour is taken from the mother or the father with equal probability *)

```
kids1 = Array[{RandomChoice[{maters[[1, 1]], maters[[paternity1[[#]], 1]]}],
0, time} &, eggs1];

kids2 = Array[{RandomChoice[{maters[[2, 1]], maters[[paternity2[[#]], 1]]}],
0, time} &, eggs2];

pop = Join[pop, kids1, kids2]]; (* Add the offspring back into the population *)
```

(* If either trading or non-trading is fixed in the population, set 'fixedQ' appropriately *)

```
If[Total[pop[[All, 1]]] == Length[pop], fixedQ = 1];
If[Total[pop[[All, 1]]] == 0, fixedQ = 0];

event++]; (* Increase the number of events by one *)
```

(* Print which strategy was fixed, the number of events until fixation, and the time until fixation *)

```
Print[fixedQ,event,time]
```