Appendix 3 - THEORY UNDERLYING THE GAME DESIGN

Introduction

Game theory provides the foundation for predicting the decisions of rational agents in strategic situations. For simple games, it is often possible to find strategic solutions in which no agent can benefit by changing their strategy (i.e., Nash equilibria). But where the possible strategy space of a game is very large (e.g. if optimal play is contingent upon dynamic local conditions such as resource distribution or game history), analytical solutions are often intractable (Hamblin 2013). To ensure sufficient realism and motivations for play, our treatments model many elephants moving independently and stochastically among spatially explicit landscape cells, and we allow for the decisions of current rounds to potentially affect payoffs in future rounds (e.g. shooting elephants subsequently reduces their number). While this critical game realism precludes us from deriving analytical solutions for optimal play, it is possible to derive analytical solutions for simplified conditions (e.g. a single round of game play and expected elephant distribution), and to explore the consequences of dominant (though not necessarily optimal) strategies (such as "always scare when elephants are on a cell, else farm") that might be used in game play.

Stakeholders in our games needed to consider the discrete movement of elephants on a spatially explicit landscape, while simultaneously considering how current decisions might affect future payoffs. Under such complex conditions, considering the full range of possible strategies available to players is not tractable, nor would it be particularly useful for understanding actual stakeholder decision making in our behavioural games. Nevertheless, it is worthwhile to relate the behavioural games being played back to first principles of game theory. In this supplementary material, we analyse a simplified version of the behavioural game from the main text and demonstrate that while farming all landscape cells is a Nash equilibrium, cooperative play to build elephant habitats can ultimately lead to higher payoffs if the temptation to defect can be avoided. We also show the payoffs associated with heuristic strategies played when elephant distributions are discrete across the landscape, and when shooting elephants can have long-term consequences on accrued payoffs in late rounds of the behavioural game. Finally, we show all R code used to analyse Nash equilibria. This supplementary material is organised as follows.

1. Nash equilibria for simplified game
2. Issues arising from elephant distributions
3. Issues arising from sequential rounds
4. Supporting code: Annotated functions

In the first section, we consider a game played for a single round, and given expected (i.e., probabilistic) rather than realised elephant distributions.

Nash equilibria for simplified game

A Nash equilibrium is a stable state of strategies for a game, from which no invading strategy can outperform the resident strategy, hence any individual player performs best by adopting the resident strategy. Below, we have developed code that allows the user to place three identical resident strategies on the simulated game landscape for any set of game parameter combinations. The test_fitness function then iterates *every possible* invading strategy and checks its fitness against the fitnesses of the resident strategy. It does this by simulating the player in the upper right corner of the game landscape (note that due to landscape symmetry, choice of landscape quadrant does not matter). In the elephant games, players can choose from four possible options for each of their nine cells:

1. Farm
2. Scare

3. Cull
4. Habitat

Each option is associated with points, a cost, and a weight that affects the cell's attractiveness to geese (and those of neighbouring cells). There are $4^9 = 262144$ possible combinations of farm, scare, cull, and habitat choice on the nine cells. Hence, to test whether or not a resident strategy can be invaded by a different strategy, we cycle through all 262144 possible land use choice combinations that could possibly invade the resident strategy . If none of these combinations results in a higher payoff than the resident strategy (i.e., if the best invading strategy *is* the resident strategy), then we have proved through exhaustive search that the resident strategy is a Nash equilibrium for the chosen game conditions.

The key simplifying assumption we make in assessing Nash equilibria is that payoffs are calculated from the expected distributions of elephants (based on landscape cell weights) rather than realised distributions of individual elephants. For example, on a landscape in which all cells are being farmed and therefore of equal weight and probability of elephant occurrence , each cell is assumed to have $18/36 = 0.5$ elephants. Where different land-use decisions are made, expected elephant numbers are adjusted accordingly by cell weights. This simplification preserves the general structure of the game and allows us to investigate it from first principles using game theory. Allowing instead for realised elephant distributions would make calculation of Nash equilibria using our method intractable, as there are $36^{18} \approx 1.03 \times 10^{28}$ possible ways that 18 elephants can be distributed across the landscape (although this number could be reduced somewhat by identifying symmetries on the landscape). It would also likely result in complex strategies, conditional upon realised elephant distributions; we explore such strategies in the following section.

The test_fitness function works by iterating through all possible invading strategies and calculating the payoff of each. If the background strategy is a Nash equilibrium, then the highest payoff score will also be the background strategy. All parameter values are included as arguments, which are listed in Table 1 of the main text, recreated below. In this simplified game, we assume no elephant habitat subsidy.

	Farm	Farm and scare	Farm and kill	Elephant habitat
Yield	4	4	4	0
Subsidy	0	0	0	X [2, 4, 6]
Crop damage (per elephant)	-2	-2	-2	0
Costs	0	-1	-2	0
Weight	10	5	2	90
Effectiveness	–	30%	80%	–
Habitat neighbourhood effect	None	None	None	+5 weight to neighbour cells

The parameter values in the table above are set as default arguments in the function test_fitness() function, which is shown below.

```
test_fitness <-  function(land,
                  farm_points     = 4,
                  scare_points    = 4,
                  cull_points     = 4,
                  habitat_points  = 0,
```

```
                farm_cost       = 0,
                scare_cost      = 1,
                cull_cost       = 2,
                habitat_cost    = 0,
                farm_weight     = 10,
                scare_weight    = 5,
                cull_weight     = 2,
                habitat_weight  = 90,
                bump            = 5,
                habitat_neigh   = 1,
                eleph_count     = 18,
                damage          = 2,
                scare_prob      = 0.8,
                cull_prob       = 0.3,
                shoot_to_kill   = TRUE,
                replace_living  = FALSE
){
  parameters <- c(farm_points, scare_points, cull_points, habitat_points,
            farm_cost, scare_cost, cull_cost, habitat_cost,
            farm_weight, scare_weight, cull_weight, habitat_weight,
            bump, habitat_neigh, eleph_count, damage,
            scare_prob, cull_prob, shoot_to_kill, replace_living);
  perms     <- expand.grid( c1 = 1:4, c2 = 1:4, c3 = 1:4, c4 = 1:4, c5 = 1:4,
                    c6 = 1:4, c7 = 1:4, c8 = 1:4, c9 = 1:4);
  tot_perms  <- dim(perms)[1];
  fit_vector <- rep(0, tot_perms);

  time_elapsed <- proc.time();
  for( strat in 1:tot_perms ){
    temp_l          <- land;
    temp_l[1,4]     <- perms[strat,1];
    temp_l[1,5]     <- perms[strat,2];
    temp_l[1,6]     <- perms[strat,3];
    temp_l[2,4]     <- perms[strat,4];
    temp_l[2,5]     <- perms[strat,5];
    temp_l[2,6]     <- perms[strat,6];
    temp_l[3,4]     <- perms[strat,7];
    temp_l[3,5]     <- perms[strat,8];
    temp_l[3,6]     <- perms[strat,9];
    temp_l          <- unlist(temp_l);
    land            <- matrix(data = temp_l, nrow = 6, ncol = 6);
    land_pay        <- calc_payoff(land, parameters);
    strat_fitness   <- sum(land_pay[1:3, 4:6]);
    fit_vector[strat] <- strat_fitness;
    time_check      <- proc.time();
    time_print      <- time_check - time_elapsed;
    if(time_print[3] > 30){
      pct_complete <- round(strat / tot_perms * 100);
      print(paste("Progress: ", pct_complete, "%", sep = ""));
      time_elapsed <- proc.time();
    }
```

```
  }

  output <- list(strategy = perms, fitness = fit_vector, land = land);

  return( output );
}
```

Note that the test_fitness function relies on the custom function calc_payoff to calculate the payoff of a focal strategy (i.e., the payoff of a focal set of land-use decisions, as played in the upper right corner of the landscape), which in turn calls several other custom functions. These custom functions are explained in detail below, but here it is only important that calc_payoff calculates the payoff of a focal invading strategy against a selected resident strategy. The loop in the above cycles through every possible invading strategy to calculate all possible payoffs. In the output of test_fitness, the list of strategies is returned (strategy), along with the fitness of each strategy (fitness; vector elements correspond to rows of strategy), and the original landscape (land).

Resident farming strategy: To show that farming on all cells is a Nash equilibrium, it is first necessary to define a landscape as a six by six matrix in which the background strategy land-use choices are being played. For farming, cell land use choice takes a value of 1, so the appropriate land is simply a matrix of 1s.

```
proposed_NE <- matrix(data = 1, nrow = 6, ncol = 6);
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    1    1    1
## [2,]    1    1    1    1    1    1
## [3,]    1    1    1    1    1    1
## [4,]    1    1    1    1    1    1
## [5,]    1    1    1    1    1    1
## [6,]    1    1    1    1    1    1
```

The land matrix is then used in the test_fitness function, where all the payoffs of all possible invading strategies are compared to the payoffs of the resident strategy.

```
fitness_results <- test_fitness(land = proposed_NE);
```

These results are then summarised with the following function fitness_summary.

```
fitness_summary <- function(results, background = NULL, plot = FALSE){
  if(is.null(background) == TRUE){
    background <- matrix(data = 1, nrow = 3, ncol = 3);
    warning("No resident strategy selected: assuming all farming");
  }
  fitness   <- results$fitness;
  strategy  <- results$strategy;
  land      <- results$land;
  fit_order <- order(fitness, decreasing = TRUE);
  top_ten   <- fit_order[1:10];
  payoff    <- fitness[top_ten];
  most_str  <- strategy[top_ten,];
  res_tabl  <- cbind(payoff, most_str);
  bgstrat   <- unlist(t(background)[1:9]);
  permpos   <- 1;
  checkstr  <- 0;
  time_elapsed <- proc.time();
  while(checkstr == 0 & permpos < dim(strategy)[1]){
```

```
    sqrdev <- (bgstrat - strategy[permpos,])*(bgstrat - strategy[permpos,]);
    if( sum(sqrdev) == 0 ){
      checkstr <- 1;
    }else{
      permpos <- permpos + 1;
    }
    time_check      <- proc.time();
    time_print      <- time_check - time_elapsed;
    if(time_print[3] > 10){
      pct_complete <- round(permpos / dim(strategy)[1] * 100);
      print(paste("Checked: ", pct_complete, "%", sep = ""));
      time_elapsed <- proc.time();
    }
  }
  last_row <- c(fitness[permpos], bgstrat);
  res_tabl <- rbind(res_tabl, last_row);
  rownames(res_tabl) <- c("Strategy 1", "Strategy 2", "Strategy 3",
                          "Strategy 4", "Strategy 5", "Strategy 6",
                          "Strategy 7", "Strategy 8", "Strategy 9",
                          "Strategy 10", "Resident Strategy");
  if(plot == TRUE){
    par(mar = c(5, 5, 1, 1), lwd = 2);
    hist(fitness, xlab = "Strategy Fitness", ylab = "Frequency",
        main = "", cex = 1.5, cex.lab = 1.5, cex.axis = 1.5, col = "grey");
  }
  return(res_tabl);
}
```

The function fitness_summary organises the results from test_fitness and generates an ordered list of invading strategies by fitness. If the highest fitness strategy is the resident strategy, then it will be the first listed in the table and the resident strategy will be a Nash equilibrium. The fitness_summary argument background is for the user to set what the equivalent 'resident' strategy looks like for the invader. The reason that the background strategy is not just assumed to be identical to the other three players is because an 'identical' strategy might actually rely on symmetry in land orientation – e.g., if everyone farms all their squares *except* the square in the middle of the board.

```
inv_bgd <- matrix(data = 1, nrow = 3, ncol = 3);
results <- fitness_summary(results    = fitness_results,
                           background =  inv_bgd);
```

Results for a resident strategy of farming all landscape cells are shown below.

		c 1	c 2	c 3	c 4	c 5	c 6	c 7	c 8	c 9
	payoff									
Strategy 1	27.00000	1	1	1	1	1	1	1	1	1
Strategy 2	26.68879	2	1	1	1	1	1	1	1	1
Strategy 3	26.68879	1	2	1	1	1	1	1	1	1
Strategy 4	26.68879	1	1	2	1	1	1	1	1	1

Strategy 5	26.68879	1 1 1 2 1 1 1 1 1
Strategy 6	26.68879	1 1 1 1 2 1 1 1 1
Strategy 7	26.68879	1 1 1 1 1 2 1 1 1
Strategy 8	26.68879	1 1 1 1 1 1 2 1 1
Strategy 9	26.68879	1 1 1 1 1 1 1 2 1
Strategy 10	26.68879	1 1 1 1 1 1 1 1 2
Resident Strategy	27.00000	1 1 1 1 1 1 1 1 1

The payoff is indicated in the second column, while c1 through c9 refer to the invading strategy's landscape cells ordered by row as below in table form.

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Given that the highest fitness strategy is the resident strategy of farming on all cells, with a total payoff of 27, we can say that farming on all cells is a Nash equilibrium strategy; if all neighbours are farming all of their cells, then the best strategy a focal player can have is to also farm all cells.

It is important to note that just because farming on all cells is a Nash equilibrium, this does not mean that farming on all cells also yields the highest payoff per player. Indeed, we can show using the same method that a cooperative strategy replacing farming with elephant habitat in each player's centre-most landscape cell yields a higher payoff for each player. Consider the landscape below, and recall that 4 indicates the choice of elephant habitat.

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    1    1    1    1
## [2,]    1    1    1    1    1    1
## [3,]    1    1    4    4    1    1
## [4,]    1    1    4    4    1    1
## [5,]    1    1    1    1    1    1
## [6,]    1    1    1    1    1    1
```

The above cooperative resident strategy yields more than 27 points, but is not a Nash equilibrium. To demonstrate this, the below code is run as before.

```
proposed_NE_coop        <- matrix(data = 1, nrow = 6, ncol = 6);
proposed_NE_coop[3:4, 3:4] <- 4;
fitness_results_coop    <- test_fitness(land = proposed_NE_coop);
inv_bgd_coop            <- matrix(data = 1, nrow = 3, ncol = 3);
inv_bgd_coop[3, 1]      <- 4; # Habitat in the lower left corner
results_coop            <- fitness_summary(results    = fitness_results_coop,
                                           background = inv_bgd_coop);
```

The below table shows the results.

	payoff	c1	c2	c3	c4	c5	c6	c7	c8	c9
Strategy 1	30.25000	1	1	1	1	1	1	1	1	1
Strategy 2	30.14320	1	1	1	1	1	1	2	1	1
Strategy 3	29.80237	1	1	1	2	1	1	1	1	1
Strategy 4	29.80237	1	1	1	1	1	1	1	2	1
Strategy 5	29.72078	1	1	1	2	1	1	2	1	1
Strategy 6	29.72078	1	1	1	1	1	1	2	2	1
Strategy 7	29.63195	2	1	1	1	1	1	1	1	1
Strategy 8	29.63195	1	2	1	1	1	1	1	1	1
Strategy 9	29.63195	1	1	2	1	1	1	1	1	1
Strategy 10	29.63195	1	1	1	1	2	1	1	1	1
Resident Strategy	27.50000	1	1	1	1	1	1	4	1	1

While the total payoff of the resident strategy has increased slightly to 27.5 from 27 when all players were farming, this cooperative use of elephant habitat is not a Nash equilibrium because the highest payoff strategy is still farming, which now yields an even higher payoff of 30.25. Hence, when the game is analysed for a single round of play given payoffs of expected elephant distributions, it entails a classic Prisoner's dilemma situation in which rational play by all agents leads to a lower payoff than would be possible through cooperation.

Resident scaring strategy: Interestingly, if the resident strategy is such that all farmers scare elephants on all cells, then the most successful invading strategy is one that scares on every cell except one, in which elephant habitat is instead provided. Using the same techniques as above, the following strategies payoffs accrue.

	payoff	c1	c2	c3	c4	c5	c6	c7	c8	c9
Strategy 1	18.71382	2	2	2	2	2	2	4	2	2
Strategy 2	18.69544	4	2	2	2	2	2	2	2	2
Strategy 3	18.54265	2	2	2	2	2	2	2	2	4

Strategy 4	18.48862	2	2	4	2	2	2	2	2	2
Strategy 5	18.47838	2	2	1	2	2	2	4	2	2
Strategy 6	18.47838	2	1	2	2	2	2	4	2	2
Strategy 7	18.47838	1	2	2	2	2	2	4	2	2
Strategy 8	18.45241	2	2	2	2	2	1	4	2	2
Strategy 9	18.42327	2	2	2	2	2	2	4	2	1
Strategy 10	18.40228	4	2	1	2	2	2	2	2	2
Resident Strategy	16.16820	2	2	2	2	2	2	2	2	2

As noted above, the top scoring strategy yields a payoff of 18.7138156, which is higher than the resident strategy of 16.1681998, meaning that scaring on all cells is not a Nash equilibrium, and can be invaded by a player who opts to set one landscape cell aside for elephant habitat. Interestingly, this strategy of scaring on all landscape cells, except for a centre-most cell, is also not a Nash equilibrium, but can itself be invaded by a strategy of scaring on only one cell and farming on the rest. The total payoff accrued to each player increases, and it is worth noting that most of the highest payoff strategies listed below are farming-centred.

	payoff	c1	c2	c3	c4	c5	c6	c7	c8	c9
Strategy 1	27.29826	1	1	1	1	1	1	2	1	1
Strategy 2	27.18349	1	1	1	1	1	1	1	1	1
Strategy 3	27.04452	1	1	1	1	1	1	2	2	1
Strategy 4	27.03048	1	1	1	2	1	1	2	1	1
Strategy 5	26.88839	1	1	1	1	1	1	1	2	1
Strategy 6	26.87459	1	1	1	2	1	1	1	1	1
Strategy 7	26.82997	1	1	1	1	1	1	2	1	2
Strategy 8	26.82295	1	1	1	1	2	1	2	1	1
Strategy 9	26.82295	1	1	1	1	1	2	2	1	1

Strategy 10	26.81441	2	1	1	1	1	1	2	1	1
Resident Strategy	22.33192	2	2	2	2	2	2	4	2	2

In the above, the background and highest fitness strategy has a payoff of 27.2982609, slightly higher than the payoff accrued to one player when all players farm. Nevertheless, this highest fitness strategy in the example above is also vulnerable to invasion, this time from our originally considered Nash equilibrium strategy of farming all cells, as is shown by the highest payoff strategy below.

	payoff	c1	c2	c3	c4	c5	c6	c7	c8	c9
Strategy 1	26.27824	1	1	1	1	1	1	1	1	1
Strategy 2	25.99745	1	1	1	1	1	1	2	1	1
Strategy 3	25.99745	1	1	1	1	1	1	1	2	1
Strategy 4	25.99745	1	1	1	1	1	1	1	1	2
Strategy 5	25.99247	2	1	1	1	1	1	1	1	1
Strategy 6	25.99247	1	2	1	1	1	1	1	1	1
Strategy 7	25.99247	1	1	2	1	1	1	1	1	1
Strategy 8	25.99247	1	1	1	2	1	1	1	1	1
Strategy 9	25.99247	1	1	1	1	2	1	1	1	1
Strategy 10	25.99247	1	1	1	1	1	2	1	1	1
Resident Strategy	25.99745	1	1	1	1	1	1	2	1	1

Hence, by induction, it is clear that a community of players who scare elephants on all cells is prone to eventual replacement by a community of farmers. A strategy in which all players scare on all cells will be invaded by a strategy in which one player scares on all but one cell (leaving elephant habitat in their centre-most cell), which in turn will be invaded by a strategy of farming all but one cell (scaring elephants in their centre-most cell), which will finally be invaded by a strategy of farming on all cells. The same occurs for a community of players who shoot elephants on all cells, which (like uniform scaring) can also be invaded by a strategy of scaring on all but one cell.

Resident shooting strategy: When the resident strategy is to shoot on all landscape cells, the highest payoff invading strategy is to scare on all cells except one where a single landscape cell of habitat is instead placed.

	payoff	c1	c2	c3	c4	c5	c6	c7	c8	c9
Strategy 1	20.96820	2	2	2	2	2	2	4	2	2
Strategy 2	20.64253	4	2	2	2	2	2	2	2	2
Strategy 3	20.64253	2	2	2	2	2	2	2	2	4
Strategy 4	20.56269	2	2	2	2	2	1	4	2	2
Strategy 5	20.56269	2	2	2	2	2	2	4	2	1
Strategy 6	20.56269	2	2	1	2	2	2	4	2	2
Strategy 7	20.56269	2	1	2	2	2	2	4	2	2
Strategy 8	20.56269	1	2	2	2	2	2	4	2	2
Strategy 9	20.38758	2	2	4	2	2	2	2	2	2
Strategy 10	20.25904	2	2	2	2	2	1	4	2	1
Resident Strategy	11.70000	3	3	3	3	3	3	3	3	3

Hence shooting on all cells is not a Nash equilibrium, while the strategy of providing habitat on one (central) cell and scaring on all of the rest is surprisingly robust.

Summary: We have proven through exhaustive search that farming all landscape cells is a Nash equilibrium in a single round of the game described in the text given expected elephant distributions (i.e., where the cost of an elephant on each cell is determined by the expected number of elephants on the cell). We have also demonstrated that a cooperative strategy allocating at least one landscape cell to elephant habitat yields a higher payoff for each player, but that this cooperative strategy can be invaded by a selfish strategy that only farms. Finally, we have shown that strategies of scaring or shooting elephants on all landscape cells are vulnerable to invasion by strategies that are more farming-focused. The important outcome of this exercise is to show that the theoretical foundation of the complex elephant game played among stakeholders in the main text is grounded by the classic situation in which rationally acting agents will play a selfish strategy despite cooperative play yielding a higher total payoff.

Using the functions test_fitness and fitness_summary, it can additionally be shown that scaring, killing, or placing elephant habitat on all cells are not Nash equilibria, with all being invaded by a 'farm all cells' strategy. Hence, for the simplified game structure, it is always best for a rational agent to farm all of their cells. It is important to emphasise that such a strategy is not necessarily rational once the assumption of expected elephant distribution is relaxed and elephants are allowed to vary stochastically across the landscape. In this case, due to chance, discrete elephants will appear on some cells and not others, and with a probability that is proportional to cell weights. Players will therefore need to decide what to do when they are faced with one or more elephants on specific

cells but not others. In this case, the number of possible ways that 18 elephants can be distributed across 36 landscape cells makes calculating the payoff consequences of different strategies for each possible elephant distribution intractable. Further, given this level of game complexity, it is highly unlikely that real human players will play completely rationally, so it is more useful to consider the consequences of heuristic strategies that yield high payoffs. We do this in the next section.

Issues arising from elephant distributions

When elephants are placed discretely on the landscape, and therefore have discrete by-cell effects on crop loss rather than expected effects proportional to their probability of occurring on a given landscape cell, game players must decide what to do with elephants found on specific cells. Rational strategies in this case will likely not correspond to specific land-use choices on landscape cells, but rather decisions about what to do upon observing ϵ elephants on a given landscape cell; this decision might be affected by the strategies of other players and the distribution of elephants on other players' lands.

Recall that the minimum cell payoff is 0, elephants are randomly and uniformly distributed across landscape cells, and multiple elephants per cell is permitted. In a single round of game play, scaring and shooting actions take immediate effect. There are two heuristic strategies that are especially worth considering, which we define as 'scare-on-cells' and 'shoot-on-cells'. In the scare-on-cells strategy, players scare on any cell containing at least one elephant, but otherwise farm. In the shoot-on-cells strategy, players shoot on any cell containing at least one elephant, but otherwise farm. Below, we discuss the consequences of each strategy for a single round of game play.

The scare-on-cells strategy. The scare-on-cells strategy is likely a useful heuristic for playing the elephant game. Elephants on a landscape cell reduce the payoff yielded from the cell by 2 (Δ). Scaring elephants comes with a cost (C_{scare}) of 1 and has a 0.8 probability of success. It therefore comes with a potential increase in payoff of 1 if there is one elephant on the cell and 3 if there are two or more elephants on the cell. In the case of a single elephant, all else being equal, the probability that the elephant will be scared onto one of the focal player's remaining 8 cells (thereby negating the benefit of the action) is roughly 0.23. Using this value, the probability of scaring to a cell of a neighbouring player is therefore $Pr(scared) \approx 0.8 \times (1 - 0.23) \approx 0.616$. In other words, this is the probability that by scaring on a cell, the elephant leaves the cell and does not return to a different cell on the focal player's landscape. All else being equal, the expected number of points accrued from scaring on a cell with ϵ elephants is as follows,

$$E_{scare}(\epsilon) = Y - C_{scare} - \Delta\epsilon(1 - Pr(scared)).$$

In the above, Y is the yield from farming on the cell. Verbally, the above therefore describes the payoff yield from farming, minus the cost of scaring, minus the damage of elephants after scaring. Scaring damage is calculated as the damage per elephant (Δ), times the number of elephants (ϵ), times the probability that an elephant is not scared successfully ($1 - Pr(scared)$). For a landscape cell containing a single elephant, expected yield is as follows,

$$E[scare_{\epsilon=1}] = 4 - 1 - 2(1)(1 - 0.616) = 2.232.$$

Note that $E[scare_{\epsilon=1}] = Yield - Cost_{scare} = 3$ when $\epsilon = 0$, but as ϵ increases, the expected number of points accrued from scaring can actually become negative. Consider the instructive though highly unlikely case in which $\epsilon = 18$ (i.e., all elephants are on a single cell). Because the minimum possible cell yield is 0, in such a situation it would be a better strategy to simply farm the cell or turn it into elephant habitat (both have a cost of 0) because scaring elephants on the cell risks dispersing all 18 of them to other cells and spreading the damage. The focal player is simply better off accepting the loss of the 3 potential yield from farming on a single cell (4 minus 1 for the cost of scaring) to ensure a yield of 4 on all of the remaining 8 cells, regardless of what other players are doing.

For illustrative purposes, now assume that there exist ϵ elephants on a particular landscape cell of interest. Further assume that all other landscape cells are farmed, and that any other elephants on the landscape can be ignored for the purpose of predicting payoffs. We can consider how low ϵ needs to be for scaring them to be beneficial for a focal player when all elephants are on a single cell. First, note that to expect to gain any points at all from the cell on which elephants are located (even ignoring $Cost_{scare}$, and the possibility of elephants being displaced to a focal player's other cells), it must be the case that $\epsilon < 10$. When $\epsilon = 10$, the number of elephants remaining on the cell is expected to be 2 ($\epsilon(1 - 0.8)$), which would still result in the minimum possible crop yield of zero. When accounting for the cost of scaring and probability that scared elephants will return to one of the focal player's own landscape cells, with the above equation, scaring is only expected to increase payoff when $\epsilon < 4$. Values of $\epsilon \geq 4$ result in a negative $E[scare]$, meaning the action should not be taken (a higher payoff would be possible by farming the cell, or by turning it into elephant habitat). Nevertheless, it should be noted that $\epsilon \geq 4$ is highly unlikely, and that this situation was very rarely observed during behavioural games.

Given that the realised number of elephants per landscape cell is rarely more than three, the heuristic strategy of scare-on-cells is generally a good one. In this case, all else being equal, scaring increases a focal player's total payoff. Next, we will investigate the shoot-on-cells heuristic strategy in more detail.

The shoot-on-cells strategy

Shooting elephants potentially removes them from the entire landscape, thereby decreasing the total number of elephants that can subsequently decrease crop yield on a focal player's landscape cells. But unlike scare-on-cells, a shoot-on-cells strategy is not very beneficial for a single round of play. The probability of successfully shooting an elephant is low ($Pr(shot) = 0.3$), and from a focal player's payoff perspective, completely removing the elephant from the landscape gives no more benefit than scaring it onto a neighbouring player's cell. Because elephants are not displaced upon shooting, calculating the expected payoff for shooting an elephant on a landscape cell is relatively straightforward,

$$E[shoot] = Y - C_{shoot} - \Delta\epsilon(1 - Pr(shot))$$

In the above, C_{shoot} is the cost of shooting. For a landscape cell containing a single elephant, expected yield is as follows,

$$E[shoot_{\epsilon=1}] = 4 - 2 - 2(1)(1 - 0.3) = 0.6.$$

In this case, the expected payoff of shooting the elephant is actually lower than simply farming the landscape cell; the cost of shooting is too high, and the probability of success is too low, for shooting to be worthwhile. When $\epsilon > 1$, $E[shoot] = 0$ regardless of whether farming or shooting is chosen (if farming, then elephant damage reduces crop yield to zero; if shooting, elephant damage is expected to reduce crop yield to 1.2, but the cost of shooting is an additional 2). Hence, shooting elephants is never beneficial in a single round of the game. In the next section, we will look at how the shoot-on-cells strategy can affect points accrued over the course of 6-8 rounds of play in behavioural games.

Issues arising from sequential rounds

In previous sections, we examined simplified versions of the behavioural game in the main text, either by using expected rather than realised spatial distributions of elephants, or by considering payoff consequences for a single round of game play. When players interact over multiple rounds of game play, the parameter space of possible strategies increases exponentially to include strategies that are conditional upon game history. These strategies could be dependent upon the actions of, and payoffs accrued by, one or more players over the course of previous game rounds (e.g., a strategy might be to act one way if some number of other players did something within the previous 3 rounds, but act a different way if not). The complexity permitted in such strategies, and the

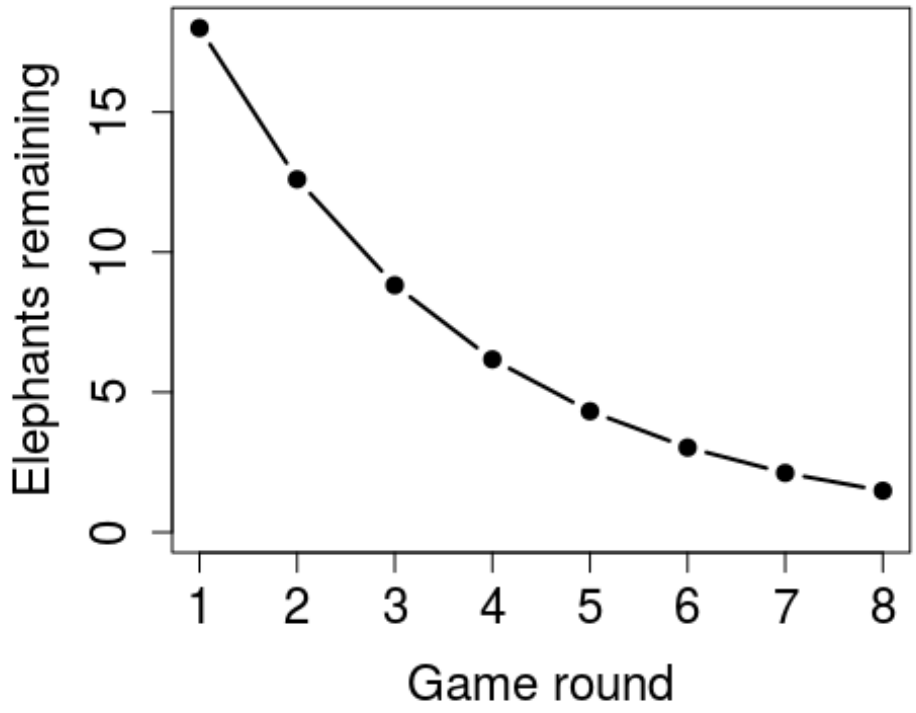
consequent challenge of assessing their costs and benefits, is illustrated by the considerable amount of literature surrounding iterative strategies for the simple Prisoner's dilemma game (Darwen and Yao 1995; Adami and Hintze 2013; Rapoport et al. 2015). We therefore cannot attempt a detailed assessment of even a fraction of the possible strategies of the behavioural games played in the main text. Instead, here we consider only the most obvious, and likely most influential, effect of game history on player strategies; when an elephant is shot, there is one fewer elephant to cause crop damage on the landscape for all subsequent rounds of play.

Long-term gains of shooting of elephants

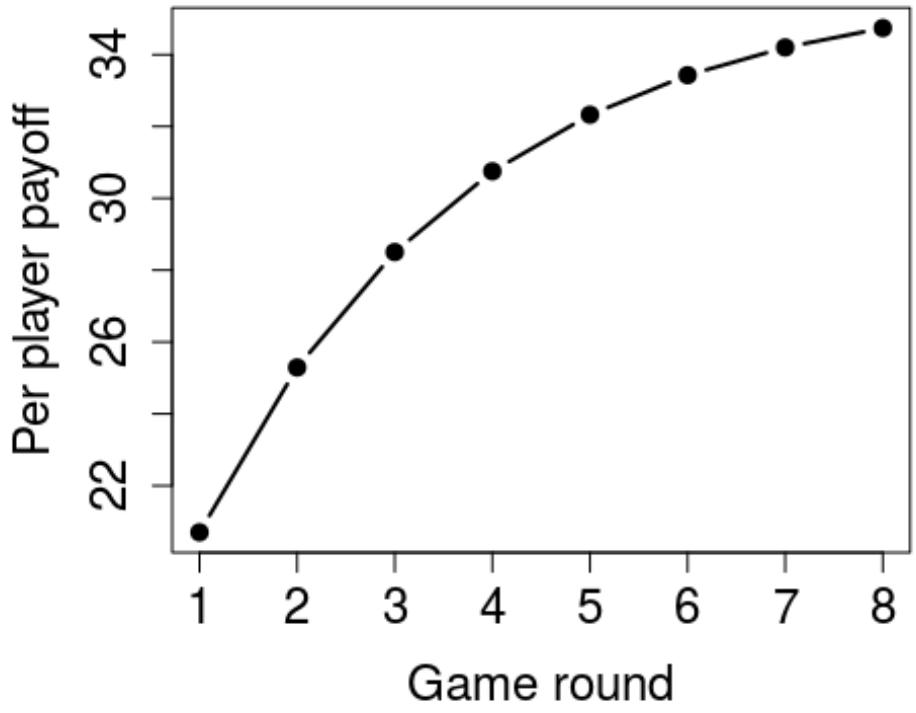
Behavioural games are played over the course of 6-8 rounds. Given this constraint, we can predict how elephant number is expected to decrease if all players shoot elephants when elephants are observed on their landscape cells. The expected number of elephants in round $r + 1$ is as follows,

$$E[\epsilon_{r+1}] = \epsilon_r(1 - Pr(shot))$$

In the behavioural games, $Pr(shot) = 0.3$, and we can plot $E[\epsilon_{r+1}]$ over rounds assuming that all players attempt to shoot elephants.



Overall, we see an exponential decrease in elephant number. By round six (in which some games terminate), the combined efforts of four players reduce expected elephant number to 3.02526. An additional two rounds brings expected elephant number down to 1.4823774. This greatly reduces the potential for elephant damage on the landscape for later rounds, but the cost of shooting in each round also needs to be considered. In each round, the expected total cost of shooting across all players will be equal to twice the number of expected elephants ($C_{shoot} = 2$), while the expected cost for a single focal player will be equal to half the number of expected elephants (assuming the expected distribution of elephants is uniform). If we restrict potential strategies to farming and shooting, we can calculate the expected payoff per player over time as elephant number decreases.



To look at the benefit of shooting over rounds, we can compare the marginal benefit accrued from shooting (i.e., the increased payoff per player above the baseline expected if no shooting had taken place) to the accrued cost of shooting (i.e., the total amount spent over rounds on shooting).

As indicated by the plot above, when all players are shooting elephants on their cells, the benefit of shooting will have outweighed the cost of shooting by round six. Any subsequent rounds 7-8 will lead to an even higher payoff associated with lower elephant number. The accrued benefit begins to outweigh the cost of shooting because with each passing round, more farmed cell yields accumulate that would not have accumulated if elephants had not already been shot, and the cost of shooting also begins to drop as fewer elephants occupy landscape cells. In other words, the early decision by players to shoot elephants can payoff in later rounds because once elephants have been eliminated, yield can be collected in higher numbers with each passing round with less need to spend costs on shooting. Were games to continue for an indefinite number of rounds with this

strategy, eventually all elephants would be eliminated, thereby increasing farm yield to its maximum per cell payoff for each cell, and eliminating the cost of shooting altogether (by eliminating the need to shoot). Hence, when round history is considered, long-term cooperative strategies of shooting can be beneficial.

Note that the above estimate for when determining when sustained shooting becomes more beneficial than costly is conservative because sometimes more than one elephant will occupy a single cell. When this happens, the cost of shooting will be reduced by two times the additional number of elephants on the cell because the cost of shooting is accrued on a per cell basis, not a per elephant basis. Also note that the same long-term rationale for shooting applies to individual players, assuming that elephants are not scared onto focal player's land. The expected per-player costs and benefits accrued over rounds will not change in this case because each player is expected to start with 4.5 elephants on their land. Relaxing this assumption, players that start with more elephants on their landscape will also accrue the long-term payoff benefits of shooting more rapidly than players that start with fewer elephants on their landscape cells.

Supporting code: Annotated functions

Here we list and explain all functions called by calc_payoff in the test_fitness function above, starting with calc_payoff itself. All functions are publicly available on GitHub. The calc_payoff function above refers to a function that calls other functions to calculate the payoff of an invading strategy.

```
calc_payoff <- function(land, parameters){
  farm_points    <- parameters[1];
  scare_points   <- parameters[2];
  cull_points    <- parameters[3];
  habitat_points <- parameters[4];
  farm_cost      <- parameters[5];
  scare_cost     <- parameters[6];
  cull_cost      <- parameters[7];
  habitat_cost   <- parameters[8];
  damage         <- parameters[16];
  eleps_distr    <- place_eleps(land, parameters); # Expected eleph number
  eleps_distr    <- disturb_eleps(land, parameters, eleps_distr);
  pay_mat        <- matrix(data = 0, nrow = dim(land)[1], ncol = dim(land));
  # Assign points to each landscape type
  pay_mat[land < 4]    <- pay_mat[land < 4] - eleps_distr[land < 4] * damage;
  pay_mat[pay_mat < 0] <- 0;
  pay_mat[land == 1]   <- pay_mat[land == 1] + farm_points
  pay_mat[land == 2]   <- pay_mat[land == 2] + scare_points
  pay_mat[land == 3]   <- pay_mat[land == 3] + cull_points
  pay_mat[land == 4]   <- pay_mat[land == 4] + habitat_points
  pay_mat[land < 4]    <- pay_mat[land < 4]  - eleps_distr[land < 4] * damage;
  pay_mat[pay_mat < 0] <- 0;
  pay_mat[land == 1]   <- pay_mat[land == 1] - farm_cost;
  pay_mat[land == 2]   <- pay_mat[land == 2] - scare_cost;
  pay_mat[land == 3]   <- pay_mat[land == 3] - cull_cost;
  pay_mat[land == 4]   <- pay_mat[land == 4] - habitat_cost;
  return(pay_mat);
}
```

This calc_payoff() function reads in the relevant parameters and the places the elephants with the place_eleps function, disturbs them (for scaring and shooting strategies), then calculates the payoffs for the landscape given the expected distribution of elephants. Note that as in the actual

behavioural game, effects of scaring or shooting take effect immediately, so an elephant on a cell at the beginning of a round might not cause crop damage to that cell if successfully scared or shot. Weights of the cells map linearly to the probability of an elephant landing on a cell, so the probability of an elephant landing on a cell is simply the cell's weight divided by the total of all cell weights. This is seen in the place_eleps function.

```
place_eleps <- function(land, parameters){
  eleph_count <- parameters[15];
  weight_mat  <- assign_cell_weight(land, parameters);
  weight_pr   <- weight_mat / sum(weight_mat);
  exp_eleps   <- eleph_count * weight_pr;
  return(exp_eleps);
}
```

The above function places elphants based on cell weight, which is assigned using the assign_cell_weight function below.

```
assign_cell_weight <- function(land, parameters){
  bump      <- parameters[13];
  rows      <- as.numeric(dim(land)[1]);
  cols      <- as.numeric(dim(land)[2]);
  weight_mat <- matrix(data = 0, nrow = rows, ncol = cols);
  weight_mat[land == 1] <- parameters[9];
  weight_mat[land == 2] <- parameters[10];
  weight_mat[land == 3] <- parameters[11];
  weight_mat[land == 4] <- parameters[12];
  weight_mat          <- habitat_bump(land, bump, weight_mat);
  return(weight_mat);
}
```

Note that each landscape option 1-4 is assigned its weight as defined in the parameters vector, then the habitat_bump function adjusts weights based on neighbourhood effects (i.e., if land == 4 indicating elephant habitat, the weight of neighbouring cells increases).

```
habitat_bump <- function(land, bump, weight_mat){
  # Early return if there's no reason to go through the ifs
  habitats <- sum(land == 4);
  if( habitats == 0 ){
    return(weight_mat);
  }
  rows        <- dim(land)[1];
  cols        <- dim(land)[2];
  for(row in 1:rows){
    for(col in 1:cols){
      if(land[row, col] == 4){
        neighbours <- get_neighbours(land, row, col);
        neighbours[neighbours == 1] <- bump;
        weight_mat <- weight_mat + neighbours;
      }
    }
  }
  return(weight_mat);
}
```

The bump is added whenever a landscape cell equals 4 indicating elephant habitat, and this bump is added to all neighbours. The neighbours of a particular cell land[row, col] is found using the get_neighbours function.

```
get_neighbours <- function(land, row, col){
  rows         <- dim(land)[1];
  cols         <- dim(land)[2];
  neighbour_mat <- matrix(data = 0, nrow = dim(land)[1], ncol = dim(land));
  if(row == 1){
    if(col == 1){
      neighbour_mat[1, 2] <- 1;
      neighbour_mat[2, 1] <- 1;
      neighbour_mat[2, 2] <- 1;
    }
    if(col == cols){
      neighbour_mat[1, col - 1] <- 1;
      neighbour_mat[2, col]     <- 1;
      neighbour_mat[2, col - 1] <- 1;
    }
    if(col > 1 & col < cols){
      neighbour_mat[1, col - 1] <- 1;
      neighbour_mat[1, col + 1] <- 1;
      neighbour_mat[2, col - 1] <- 1;
      neighbour_mat[2, col]     <- 1;
      neighbour_mat[2, col + 1] <- 1;
    }
  }
  if(row == rows){
    if(col == 1){
      neighbour_mat[rows - 1, 1] <- 1;
      neighbour_mat[rows - 1, 2] <- 1;
      neighbour_mat[rows, 2]     <- 1;
    }
    if(col == cols){
      neighbour_mat[rows - 1, col]     <- 1;
      neighbour_mat[rows - 1, col - 1] <- 1;
      neighbour_mat[rows, col - 1]     <- 1;
    }
    if(col > 1 & col < cols){
      neighbour_mat[rows, col - 1] <- 1;
      neighbour_mat[rows, col + 1] <- 1;
      neighbour_mat[rows - 1, col - 1] <- 1;
      neighbour_mat[rows - 1, col]     <- 1;
      neighbour_mat[rows - 1, col + 1] <- 1;
    }
  }
  if(row > 1 & row < rows){
    if(col == 1){
      neighbour_mat[row - 1, 1] <- 1;
      neighbour_mat[row + 1, 1] <- 1;
      neighbour_mat[row - 1, 2] <- 1;
      neighbour_mat[row, 2]     <- 1;
```

```
      neighbour_mat[row + 1, 2] <- 1;
    }
    if(col == cols){
      neighbour_mat[row - 1, col]     <- 1;
      neighbour_mat[row + 1, col]     <- 1;
      neighbour_mat[row - 1, col - 1] <- 1;
      neighbour_mat[row, col - 1]     <- 1;
      neighbour_mat[row + 1, col - 1] <- 1;
    }
    if(col > 1 & col < cols){
      neighbour_mat[row - 1, col - 1] <- 1;
      neighbour_mat[row - 1, col]     <- 1;
      neighbour_mat[row - 1, col + 1] <- 1;
      neighbour_mat[row, col - 1]     <- 1;
      neighbour_mat[row, col + 1]     <- 1;
      neighbour_mat[row + 1, col - 1] <- 1;
      neighbour_mat[row + 1, col]     <- 1;
      neighbour_mat[row + 1, col + 1] <- 1;
    }
  }
  return(neighbour_mat);
}
```

Loops are avoided in the above function to increase computational efficiency. An additional call to the place_eleps() function is used whenever elephants are scared or culled off of a cell to redistribute those displaced elephants

```
disturb_eleps <- function(land, parameters, eleps_distr){
  starting_eleps <- parameters[15];
  scare_prob     <- parameters[17];
  cull_prob      <- parameters[18];
  replace_living <- parameters[20];
  rows           <- dim(land)[1];
  cols           <- dim(land)[2];
  for(row in 1:rows){
    for(col in 1:cols){
      if(land[row, col] == 2){
        eleps_on_cell       <- eleps_distr[row, col];
        stay_prob           <- (1 - scare_prob);
        eleps_distr[row, col] <- eleps_on_cell * stay_prob;
        parameters[15]      <- eleps_on_cell * scare_prob;
        added_eleps         <- place_eleps(land, parameters);
        eleps_distr         <- eleps_distr + added_eleps;
      }
      if(land[row, col] == 3 & parameters[19] == FALSE){
        eleps_on_cell       <- eleps_distr[row, col];
        stay_prob           <- (1 - cull_prob);
        eleps_distr[row, col] <- eleps_on_cell * stay_prob;
        parameters[15]      <- eleps_on_cell * cull_prob;
        added_eleps         <- place_eleps(land, parameters);
        eleps_distr         <- eleps_distr + added_eleps;
      }
      if(land[row, col] == 3 & parameters[19] == TRUE){
```

```
            eleps_on_cell       <- eleps_distr[row, col];
            stay_prob           <- (1 - cull_prob);
            eleps_distr[row, col] <- eleps_on_cell * stay_prob;
            culled              <- eleps_on_cell * cull_prob
            parameters[15]      <- parameters[15] - culled;
            starting_eleps      <- starting_eleps - culled;
          }
        }
    }
    parameters[15] <- starting_eleps;
    if(replace_living == TRUE){
      eleps_distr <- place_eleps(land, parameters);
    }
    return(eleps_distr);
}
```

References

Adami, C., and A. Hintze. 2013. Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. Nature communications 4:2193. Nature Publishing Group.

Darwen, P. J., and X. Yao. 1995. On evolving robust strategies for iterated prisoner's dilemma. Pp. 276–292 *in* Progress in evolutionary computation.

Rapoport, A., D. A. Seale, and A. M. Colman. 2015. Is tit-for-tat the answer? on the conclusions drawn from axelrod's tournaments. PLoS ONE 10:1–11.