Supplementary Material 1: Genetic algorithm and simulations

**This supplementary material to the paper, “Effects of stakeholder empowerment on crane population and agricultural production” provides additional details concerning the use of the** [genetic algorithm](#ga) **from the** [R package GMSE](https://cran.r-project.org/package=GMSE) **v0.6.0.4, and** [code for reproducing simulations](#code)**.**

Modelling details of the genetic algorithm

Genetic algorithms are useful heuristic tools that mimic the process of biological evolution to find solutions to highly complex problems (Hamblin [2013](#ref-Hamblin2013)). In our simulations, these complex problems refer to modelling the goal-oriented decision making of managers and farmers. Managers must attempt to use their available budget to set costs that keep crane populations near a pre-specified target, while farmers must attempt to use their available budget and any actions available to maximise agricultural production. Decisions within the model are made under uncertainty, as is realistic for stakeholders in empirical social-ecological systems, and behaviour is consequently heuristic often suboptimal. Full details and explanation for how the genetic algorithm is used to model manager and stakeholder decision making are available in the GMSE package vignette (Duthie et al. [2018](#ref-Duthie2018)), [“The Genetic algorithm of GMSE”](https://cran.r-project.org/web/packages/GMSE/vignettes/SI1.pdf), which is also accessible on the [GMSE website](https://confoobio.github.io/gmse/articles/SI1.html). Here we highlight specific parameter values used in GMSE for simulated agents (managers and farmers).

The genetic algorithm of GMSE is parameterised using up to nine arguments to the gmse function used for simulation, including ga\_popsize, ga\_mingen, ga\_seedrep, ga\_sampleK, ga\_chooseK, ga\_mutation, ga\_crossover, ga\_converge\_crit, and group\_think. Details on each parameter are provided in the [GMSE documentation](https://confoobio.github.io/gmse/reference/gmse.html) and [GMSE vignettes](https://confoobio.github.io/gmse/articles/SI1.html#choosing-genetic-algorithm-parameter-values-1). Every time that an agent makes a decision (i.e., each time step), ga\_popsize temporary agent decisions are created, each of which includes a set of decisions that could potentially be made by the agent (e.g., the number of scares or culls for a farmer). Of these ga\_popsize agent decisions, ga\_seedrep are replicates of the decisions made in the previous time step of the simulation by the agent (assuming the simulation is not in its first time step). The remaining ga\_popsize - ga\_seedrep are randomly generated sets of decisions. These ga\_popsize decision sets crossover (i.e., swap values) with a probability of ga\_crossover, and mutate (i.e., change values randomly) with a probabilit of ga\_mutation. Where crossover or mutation causes a set of decisions that would be over the agent’s budget, actions are removed at random until the budget constraints are satisfied. Next, the fitness of all ga\_popsize decisions sets is evaluated against how successful it is projected to be at fulfiling the agent’s objectives (i.e., managing the population or maximising agricultural yield). A random subset of ga\_sampleK decision sets is then selected with replacement, and the ga\_chooseK highest fitness decision sets are chosen; choosing of decision sets continues in this fashion until ga\_popsize new decision sets are obtained. Once these ga\_popsize new decision sets are obtained, the genetic algorithm either terminates or continues (i.e., starts a new iteration of crossover, mutation, and selection with the new ga\_popsize decision sets). It continues if either the percent increase in highest fitness decision set is greater than ga\_converg\_crit, or if fewer than ga\_mingen iterations of crossover, mutation, and selection have passed. Upon termination, the highest fitness of the ga\_popsize is selected to become the agent’s decision-making for the time step.

In a single time step, it is possible to set group\_think = TRUE, in which case every farmer will make the same set of decisions (requiring only one call to the genetic algorithm for all farmers). For all simulations except those of scenario *a*, we used the default group\_think = FALSE. Since farmers cannot take any actions in scenario *a*, we set group\_think = TRUE to lower simulation time. Remaining parameters were set as below.

|  |  |  |
| --- | --- | --- |
| GMSE argument | Value | Comment |
| ga\_popsize | 100 | Set to the GMSE default value |
| ga\_mingen | 200 | Increased from GMSE default 40 to improve decision-making precision |
| ga\_seedrep | 20 | Set to the GMSE default value |
| ga\_sampleK | 20 | Set to the GMSE default value |
| ga\_chooseK | 2 | Set to the GMSE default value |
| ga\_mutation | 0.1 | Set to the GMSE default value |
| ga\_crossover | 0.1 | Set to the GMSE default value |
| ga\_converge\_crit | 0.01 | Decreased from GMSE default 0.1 to improve decision-making precision |

GMSE defaults were mostly used, as these have performed well in exploratory simulations and GMSE package development (Duthie et al. [2018](#ref-Duthie2018)). Where deviations from default values were chosen, it was to increase the ability of the genetic algorithm to find adaptive decision-making solutions.

Code for replicating simulations

To replicate the simulations of this manuscript, it is necessary to install the GMSE package [version 0.6.0.4](https://github.com/ConFooBio/gmse/tree/v0.6.0.4). As of the time of writing, this is the most recent version of GMSE [available on CRAN](https://cran.r-project.org/web/packages/GMSE/), and can be downloaded in R as below.

install.packages(GMSE);

GMSE v0.6.0.4 can also be installed directly from GitHub using the devtools packages (Wickham et al. [2020](#ref-devtools)).

devtools::install\_github("ConFooBio/GMSE", ref = "v0.6.0.4");

Results from the gmse function are summarised with a custom function summarise\_cranes, which is described below.

# Custom function to get crane results  
summarise\_cranes <- function(sim){  
 mat <- matrix(data = 0, nrow = length(sim$resource), ncol = 9);  
 colnames(mat) <- c("generation", "user\_budget", "pop\_size", "yield\_mn",   
 "yield\_sd", "scare\_c", "cull\_c", "scares", "culls");  
 for(i in 1:length(sim$resource)){  
 mat[i, 1] <- i;  
 mat[i, 2] <- sim$paras[i, 98];  
 mat[i, 3] <- dim(sim$resource[[i]])[1];  
 mat[i, 4] <- mean(sim$agents[[i]][sim$agents[[i]][,2] > 0, 16]);  
 mat[i, 5] <- sd(sim$agents[[i]][sim$agents[[i]][,2] > 0, 16]);  
 mat[i, 6] <- sim$action[[i]][3, 8, 1];  
 mat[i, 7] <- sim$action[[i]][3, 9, 1];  
 mat[i, 8] <- sum(sim$action[[i]][1, 8, ]);  
 mat[i, 9] <- sum(sim$action[[i]][1, 9, ]);  
 }  
 return(mat);  
}

Simulations were run using custom written functions scenario\_a, scenario\_b, scenario\_c, and scenario\_d. Each function took a vector of user budgets as an argument, which was set as the variable budgets.

budgets <- seq(from = 50, to = 4000, by = 50);

A file named abfile also specified the location of initialised crane abundances, as explained in the main text.

abfile <- "starting\_values.csv";

The contents of abfile are the 100 starting abundances for simulations, which are printed below.

print(abundances[,2]);

## [1] 18111 18167 18265 17873 17768 19138 18332 17002 18475 17500 18040 18106  
## [13] 19182 18011 17524 17618 18310 18281 17867 18270 18030 18264 17925 17703  
## [25] 17870 18589 18512 18868 18532 18081 17133 17176 17984 17760 19020 18581  
## [37] 18872 17777 18233 18648 17130 17169 17830 18206 18205 18657 17695 17139  
## [49] 18646 18321 17088 18590 17584 18733 17557 18290 19680 18702 18054 17120  
## [61] 17173 17015 17500 18818 18337 18316 18181 17977 17536 17809 17197 17297  
## [73] 18734 18515 17280 19607 18723 17292 18948 18527 18126 18008 16908 18001  
## [85] 17338 17589 17607 17631 18022 18682 18362 16857 17390 17823 18264 18309  
## [97] 18403 18720 16881 17942

A single replicate of simulation of scenario *a* is run using the code below.

# Scenario a function  
scenario\_a <- function(user\_budgets, filename, abund\_file){  
 starting\_table <- read.csv(file = abund\_file);  
 for(budget in user\_budgets){  
 ini\_abundance <- sample(x = starting\_table[,2], size = 1);  
 sim <- gmse(manager\_budget = 1000, user\_budget = budget,   
 manage\_target = 15000, RESOURCE\_ini = ini\_abundance,   
 plotting = FALSE, stakeholders = 50, times\_feeding = 10,  
 land\_ownership = TRUE, public\_land = 0.0, remove\_pr = 0,  
 scaring = FALSE, culling = FALSE, lambda = 0.118,  
 time\_max = 30, res\_death\_type = 1, res\_movement = 4,  
 res\_consume = 0.02, res\_birth\_K = 200000, observe\_type = 0,   
 agent\_view = 10, max\_ages = 20, usr\_budget\_rng = 0,   
 converge\_crit = 0.01, ga\_mingen = 200, age\_repr = 4,  
 group\_think = TRUE); # Since no actions, group\_think okay.  
 s\_mat <- summarise\_cranes(sim);  
 write.table(s\_mat, filename, sep = ",", row.names = FALSE,  
 col.names = !file.exists(filename), append = TRUE);  
 rm(sim);  
 rm(s\_mat);  
 gc();  
 }  
 return("Done!");  
}  
# Run scenario a  
scenario\_a(user\_budgets = budgets, filename = "out.csv", abund\_file = abfile);

A single replicate of simulation of scenario 2 is run using the code below.

# Scenario b function  
scenario\_b <- function(user\_budgets, filename, abund\_file){  
 starting\_table <- read.csv(file = abund\_file);  
 for(budget in user\_budgets){  
 ini\_abundance <- sample(x = starting\_table[,2], size = 1);  
 sim <- gmse(manager\_budget = 1000, user\_budget = budget,   
 manage\_target = 100000, RESOURCE\_ini = ini\_abundance,   
 plotting = FALSE, stakeholders = 50, times\_feeding = 10,  
 land\_ownership = TRUE, public\_land = 0.0, remove\_pr = 0,  
 scaring = TRUE, culling = TRUE, lambda = 0.118,  
 time\_max = 30, res\_death\_type = 1, res\_movement = 4,  
 res\_consume = 0.02, res\_birth\_K = 200000, observe\_type = 0,   
 agent\_view = 10, max\_ages = 20, usr\_budget\_rng = 0,   
 converge\_crit = 0.01, ga\_mingen = 200, age\_repr = 4,  
 group\_think = FALSE);  
 s\_mat <- summarise\_cranes(sim);  
 write.table(s\_mat, filename, sep = ",", row.names = FALSE,  
 col.names = !file.exists(filename), append = TRUE);  
 rm(sim);  
 rm(s\_mat);  
 gc();  
 }  
 return("Done!");  
}  
# Run scenario b  
scenario\_b(user\_budgets = budgets, filename = "out.csv", abund\_file = abfile);

A single replicate of simulation of scenario 3 is run using the code below.

# Scenario c function  
scenario\_c <- function(user\_budgets, filename, abund\_file){  
 starting\_table <- read.csv(file = abund\_file);  
 for(budget in user\_budgets){  
 ini\_abundance <- sample(x = starting\_table[,2], size = 1);  
 sim <- gmse(manager\_budget = 1000, user\_budget = budget,   
 manage\_target = 100000, RESOURCE\_ini = ini\_abundance,   
 plotting = FALSE, stakeholders = 50, times\_feeding = 10,   
 land\_ownership = TRUE, public\_land = 0.0, remove\_pr = 0,  
 scaring = FALSE, culling = TRUE, lambda = 0.118,   
 time\_max = 30, res\_death\_type = 1, res\_movement = 4,   
 res\_consume = 0.02, res\_birth\_K = 200000, observe\_type = 0,   
 agent\_view = 10, max\_ages = 20, usr\_budget\_rng = 0,   
 converge\_crit = 0.01, ga\_mingen = 200, age\_repr = 4,  
 group\_think = FALSE);  
 s\_mat <- summarise\_cranes(sim);  
 write.table(s\_mat, filename, sep = ",", row.names = FALSE,  
 col.names = !file.exists(filename), append = TRUE);  
 rm(sim);  
 rm(s\_mat);  
 gc();  
 }  
 return("Done!");  
}  
# Run scenario c  
scenario\_c(user\_budgets = budgets, filename = "out.csv", abund\_file = abfile);

A single replicate of simulation of scenario 4 is run using the code below.

# Scenario d function  
scenario\_d <- function(user\_budgets, filename, abund\_file){  
 starting\_table <- read.csv(file = abund\_file);  
 for(budget in user\_budgets){  
 ini\_abundance <- sample(x = starting\_table[,2], size = 1);  
 sim <- gmse(manager\_budget = 1000, user\_budget = budget,   
 manage\_target = 15000, RESOURCE\_ini = ini\_abundance,   
 plotting = FALSE, stakeholders = 50, times\_feeding = 10,   
 land\_ownership = TRUE, public\_land = 0.0, remove\_pr = 0,  
 scaring = TRUE, culling = TRUE, lambda = 0.118,   
 time\_max = 30, res\_death\_type = 1, res\_movement = 4,   
 res\_consume = 0.02, res\_birth\_K = 200000, observe\_type = 0,   
 agent\_view = 10, max\_ages = 20, usr\_budget\_rng = 0,   
 converge\_crit = 0.01, ga\_mingen = 200, age\_repr = 4,  
 group\_think = FALSE);  
 s\_mat <- summarise\_cranes(sim);  
 write.table(s\_mat, filename, sep = ",", row.names = FALSE,  
 col.names = !file.exists(filename), append = TRUE);  
 rm(sim);  
 rm(s\_mat);  
 gc();  
 }  
 return("Done!");  
}  
# Run scenario d  
scenario\_d(user\_budgets = budgets, filename = "out.csv", abund\_file = abfile);

To vary user budgets, scenarios 1-4 can be run with usr\_budget\_rng = 50 instead of usr\_budget\_rng = 0.

Literature Cited

Duthie, A. B., J. J. Cusack, I. L. Jones, E. B. Nilsen, R. A. Pozo, O. S. Rakotonarivo, B. V. Moorter, and N. Bunnefeld. 2018. GMSE: an R package for generalised management strategy evaluation. Methods in Ecology and Evolution, doi: [10.1101/221432](https://doi.org/10.1101/221432).

Hamblin, S. 2013. On the practical usage of genetic algorithms in ecology and evolution. Methods in Ecology and Evolution 4:184–194.

Wickham, H., J. Hester, and W. Chang. 2020. Devtools: Tools to make developing r packages easier.