GMSE European hamster model

Imke Tomsin

2023-2024

# Defining the different months that will be simulated

TMAX <- 60; # Maximum number of months in the simulation  
JAN <- seq(from = 1, to = TMAX, by = 12); # Time steps in JAN, FEB, MAR, etc.  
FEB <- seq(from = 2, to = TMAX, by = 12);  
MAR <- seq(from = 3, to = TMAX, by = 12);  
APR <- seq(from = 4, to = TMAX, by = 12);  
MAY <- seq(from = 5, to = TMAX, by = 12);  
JUN <- seq(from = 6, to = TMAX, by = 12);  
JUL <- seq(from = 7, to = TMAX, by = 12);  
AUG <- seq(from = 8, to = TMAX, by = 12);  
SEP <- seq(from = 9, to = TMAX, by = 12);  
OCT <- seq(from = 10, to = TMAX, by = 12);  
NOV <- seq(from = 11, to = TMAX, by = 12);  
DEC <- seq(from = 12, to = TMAX, by = 12)  
# This way, we are able to use each time step in the GMSE loop to identify the month

# Function to simulate carrying capacity by cell

cell\_K <- function(res, DIM\_1 = 447, DIM\_2 = 447){  
 for(i in 1:DIM\_1){  
 for(j in 1:DIM\_2){  
 sum\_ij <- sum(res[, 5] == i & res[, 6] == j)  
 if(sum\_ij > 1){  
 on\_ij <- which(res[, 5] == i & res[, 6] == j);  
 loser\_ij <- sample(x = on\_ij, size = sum\_ij - 1);  
 res[loser\_ij, 7] <- 0; # Stop in tracks  
 res[loser\_ij, 9] <- 1; # Definite death  
 res[loser\_ij, 10] <- 0; # No birth  
 res[loser\_ij, 11] <- 0; # No birth  
 }  
 }  
 }  
 return(res);  
}  
  
#Now, if two hamsters occupy the same cell, one of them will die.   
  
#This should be run after movement has happened, so that offspring are not removed before  
#reproductively active/old enough to leave the mother  
#This function is quite slow -- probably want to speed it up somehow

# Function to add a crop type to each cell of the landscape

# The 'land' in GMSE is a 3D array with the farm number in layer 3, the crop amount in layer 2, and nothing in layer 1. The code below redefines layer 1 by putting a crop type (1, 2 or 3) down for each farm number (you can change these as desired).The code assigns a random crop type to the land owned by each stakeholder. The function has to be run before looping over time.   
  
crop\_type <- function(land, agents = 11, types = 3){ # agents = manager + number of stakeholders, types = number of possible crop types  
 xdim <- dim(land)[1];  
 ydim <- dim(land)[2];  
 for(i in 1:xdim){   
 for(j in 1:ydim){  
 if (land[i, j, 3] %in% 1:agents) { #1:number of farms/stakeholders   
 land[i, j, 1] <- sample(1:types, 1) #1:number of possible crop types, each cell gets a different one with this function  
 }  
 }  
 }   
 return(land);  
}

# Alternative functions to assign a fixed number of (random) stakeholders/farms a certain crop type

To assign a certain crop type to a certain amount of fields, and assign the other fields a random type:

crop\_type <- function(land, agents = 11, types = 2) { #types = amount of crop types of which no fixed number is assigned to fields  
 xdim <- dim(land)[1]  
 ydim <- dim(land)[2]  
   
 type3\_stakeholders <- sample(2:agents, 5) # Select 5 random stakeholders to get assigned crop type 3  
   
 # Create a vector to store the crop type for each stakeholder  
 stakeholder\_types <- rep(NA, agents)  
   
 for (stakeholder in 2:agents) {  
 if (stakeholder %in% type3\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 3  
 } else {  
 stakeholder\_types[stakeholder] <- sample(1:types, 1)  
 }  
 }  
   
 for (i in 1:xdim) {   
 for (j in 1:ydim) {  
 if (land[i, j, 3] %in% 2:agents) {  
 land[i, j, 1] <- stakeholder\_types[land[i, j, 3]]  
 }  
 }   
 }  
 return(land)  
}

To assign a certain number of fields a certain crop type

crop\_type2 <- function(land, agents = 11) { # agents = 1 (manager) + number of stakeholders  
 xdim <- dim(land)[1]  
 ydim <- dim(land)[2]  
   
 # Randomly assign stakeholders to crop types  
 stakeholders\_vector <- sample(2:agents) #randomly shuffle the stakeholders (agents), but leave out the first (= always the manager)  
 type3\_stakeholders <- stakeholders\_vector[1:2] #2 get type 3  
 type2\_stakeholders <- stakeholders\_vector[3:6] #4 get type 2  
 type1\_stakeholders <- stakeholders\_vector[7:10] #4 get type 1  
   
 # Create a vector to store the crop type for each stakeholder  
 stakeholder\_types <- rep(NA, agents)  
   
 for (stakeholder in 2:agents) {   
 if (stakeholder %in% type3\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 3  
 } else if (stakeholder %in% type2\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 2  
 } else if (stakeholder %in% type1\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 1  
 }  
 }  
  
 for (i in 1:xdim) {   
 for (j in 1:ydim) {  
 if (land[i, j, 3] %in% 2:agents) {  
 land[i, j, 1] <- stakeholder\_types[land[i, j, 3]]  
 }  
 }   
 }  
 return(land)  
}

# Functions to change population parameters based on crop type

**Crop type 1**: crop unsuitable for hamsters. These are crops in which hamsters do not tend to occur at all, e.g. because of lack of food or coverage. In the model, hamsters have a survival probability of 0 when they establish in these fields.

**Crop type 2**: intensive agriculture of a hamster-friendly crop. These are crops that are consumed by hamsters, and in which they can survive. But, they are monocultures, and no hamster-friendly adjustments have been made to the way they are cultivated. Example: cereals that are harvested too early, leading to 0% coverage while the hamsters are still reproducing and not yet hibernating.

**Crop type 3**: Hamster-friendly crops, with adjustments for hamster-friendly agricultural practices. Later harvests allow for an additional litter

Not in here yet but I might want to add it, crop type 4: Hamster heaven. No or very late harvest, with vegetation being a mix of hamster-friendly crops and herbs etc. leading to a higher lambda.

# The function below checks each hamster's cell location, then checks the crop type on that location, and adjusts population parameters accordingly  
  
# The function currently changes mortality for hamsters in crop types 1 on a monthly basis, and changes lambda for hamsters in crop type 3 in the month August (= extra litter)  
  
crop\_hamster <- function(land, res){   
 N <- dim(res)[1]; # Number of hamsters  
 for(i in 1:N) {   
 xloc <- res[i, 5] +1; #+1 to fix indexing difference problem between R and C (otherwise error occurs about the landscape starting in (0,0))  
 yloc <- res[i, 6] +1;  
 crop <- land[xloc, yloc, 1];  
   
 # cat("Hamster ", i, " at (", xloc, ", ", yloc, ") has crop type: ", crop, "\n") #added this bit to check if the function is working properly, this shows that some hamsters' xloc or yloc randomly seems to change by 1 cell between simulations, even when movement is not simulated, sometimes both change, sometimes just one changes, and sometimes the crop type value (1, 2 or 3) also changes  
   
 if (crop == 1) {  
 res[i, 9] <- 1 # column 9 = death probability -> 100% mortality  
 #} else if (crop == 2 && next\_time %in% c(AUG,SEP)) {  
 #res[i, 9] <- res[i, 9] + 0.05 # increase mortality by 5% in months after harvest   
 } else if (crop == 3 && next\_time %in% c(JUN, JUL, AUG)) {   
 res[i, 10] <- 1.28  
 } else {  
 # Default case, nothing changes  
 }  
 }  
 return(res);  
}

# GMSE simulations

## 1: Simulation for a certain set of parameters, no changes are made to the landscape.

2 litters, in June and July, movement in June and July, starting X of 15, 30, and 250 females (last one based on 50/500 rule) (no changes to landscape/only one crop type at a time)

library(GMSE)  
DIM\_1 <- 447; # Land dimension 1  
DIM\_2 <- 447; # Land dimension 2  
  
iterations <- 100 #number of iterations  
results <- list() #list to store results from each repetition  
  
for (rep in 1:iterations) {  
 tryCatch({  
  
# Initialise the first output  
sim\_old <- gmse\_apply(stakeholders = 1, # not important here as landscape is all the same and no actions are being taken  
 res\_movement = 100,   
 remove\_pr = 1 - 0.976083968,   
 lambda = 0,   
 res\_death\_type = 1,  
 observe\_type = 2, # but only once a year  
 res\_move\_obs = FALSE,   
 max\_ages = 24,   
 RESOURCE\_ini = 250, # N   
 culling = FALSE,  
 land\_ownership = TRUE,  
 age\_repr = 4,  
 land\_dim\_1 = DIM\_1,  
 land\_dim\_2 = DIM\_2,  
 manage\_target = 2500,  
 res\_move\_type = 0,  
 get\_res = 'Full');  
old\_obs <- sim\_old[["observation\_array"]];  
  
# Initialise the results matrix for the current iteration; Currently, 2 variables are recorded as output  
# Double-brackets: technically a bit more secure. It works if you edit columns of the resource\_array directly.  
sim\_sum\_1 <- matrix(data = NA, nrow = TMAX, ncol = 2);  
colnames(sim\_sum\_1) <- c("Time", "Pop\_size");  
moving\_month <- 0;  
  
for(time\_step in 1:TMAX){  
 sim\_new <- gmse\_apply(old\_list = sim\_old, get\_res = 'Full',  
 res\_move\_type = moving\_month);  
 sim\_sum\_1[time\_step, 1] <- time\_step;  
 sim\_sum\_1[time\_step, 2] <- sim\_new$basic\_output$resource\_results[1];  
   
 next\_time <- time\_step + 1; # What happens next time step?  
 if(next\_time %in% JAN){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.976083968; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% FEB){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.94824969; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% MAR){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.9023544; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% APR){  
 moving\_month <- 1; # A moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.811569975; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% MAY){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.805694059; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 }  
 if(next\_time %in% JUN){  
 moving\_month <- 1; # A moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.794513672; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 3; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% JUL){  
 moving\_month <- 1; # A moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.767700722; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 3; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 }  
 if(next\_time %in% AUG){  
 moving\_month <- 1; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.842305069; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 3; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 }  
 if(next\_time %in% SEP){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.877005333; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 }  
   
 if(next\_time %in% OCT){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.891729601; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% NOV){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.939895976; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 if(next\_time %in% DEC){  
 moving\_month <- 0; # Not a moving month  
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.956833861; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 }  
 sim\_old <- sim\_new; # BD: This should always go at the end  
   
 #print(sim\_sum\_1[time\_step,]); #Activate to see the simulation progress  
}  
   
}, error = function(e) { #tryCatch: if error occurs because N = 0  
 cat("Error occurred in iteration", rep, ":", conditionMessage(e), "\n")  
}) #end of tryCatch  
  
  
sim\_sum\_1 <- cbind(rep, sim\_sum\_1); # Add iteration number as a column  
print(sim\_sum\_1);   
results[[rep]] <- sim\_sum\_1  
  
}  
  
combined\_results <- do.call(rbind, results)  
combined\_results[is.na(combined\_results)] <- 0  
  
#now timestep is also recorded as a 0 from the moment extinction occurred, we want time step to be recorded correctly  
  
combined\_results <- as.data.frame(combined\_results)  
  
# Loop through each repetition  
for (rep\_id in unique(combined\_results$rep)) {  
 # Subset data for the current repetition  
 rep\_data <- combined\_results[combined\_results$rep == rep\_id, ]  
   
 # Loop through each row in the current repetition's data  
 for (i in 2:nrow(rep\_data)) {  
 # If the current time step is 0, replace it with the previous time step + 1  
 if (rep\_data$Time[i] == 0) {  
 rep\_data$Time[i] <- rep\_data$Time[i - 1] + 1  
 }  
 }  
   
 # Replace the data for the current repetition in the original data frame  
 combined\_results[combined\_results$rep == rep\_id, ] <- rep\_data  
}  
  
file\_path <- "C:/Users/imket/Documents/GitHub/European\_hamster\_GMSE/Output/Simulation8\_100it\_250startN.csv" #Give the simulation an appropriate name  
write.csv(combined\_results, file = file\_path, row.names = FALSE)

## 2: Simulations modelling multiple different crop types on the landscape. Crop type influences the population parameters of the hamsters ocurring in it.

library(GMSE)  
  
#parameters specified below are for hamsters in crop type 2. For type 3 an extra litter is modelled, for type 1 a different mortality is modelled (no survival)   
  
DIM\_1 <- 447; # Land dimension 1  
DIM\_2 <- 447; # Land dimension 2  
  
iterations <- 100 #number of iterations  
results <- list() #list to store results from each repetition  
  
for (rep in 1:iterations) {  
 tryCatch({  
   
################################### Adding the landscape function here = different landscape for each iteration   
crop\_type2 <- function(land, agents = 11) { # agents = 1 (manager) + number of stakeholders  
 xdim <- dim(land)[1]  
 ydim <- dim(land)[2]  
   
 # Randomly assign stakeholders to crop types  
 stakeholders\_vector <- sample(2:agents) #randomly shuffle the stakeholders (agents), but leave out the first (= always the manager)  
 type3\_stakeholders <- stakeholders\_vector[1:2] #2 get type 3  
 type2\_stakeholders <- stakeholders\_vector[3:6] #4 get type 2  
 type1\_stakeholders <- stakeholders\_vector[7:10] #4 get type 1  
   
 # Create a vector to store the crop type for each stakeholder  
 stakeholder\_types <- rep(NA, agents)  
   
 for (stakeholder in 2:agents) {   
 if (stakeholder %in% type3\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 3  
 } else if (stakeholder %in% type2\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 2  
 } else if (stakeholder %in% type1\_stakeholders) {  
 stakeholder\_types[stakeholder] <- 1  
 }  
 }  
  
 for (i in 1:xdim) {   
 for (j in 1:ydim) {  
 if (land[i, j, 3] %in% 2:agents) {  
 land[i, j, 1] <- stakeholder\_types[land[i, j, 3]]  
 }  
 }   
 }  
 return(land)  
}  
#####################################################  
  
# Initialise the first output  
sim\_old <- gmse\_apply(stakeholders = 10, # should be the same as in function crop\_type!  
 res\_movement = 100,   
 res\_move\_type = 0,  
 remove\_pr = 1 - 0.976083968,   
 lambda = 0,   
 res\_death\_type = 1,  
 observe\_type = 2, # but only once a year  
 res\_move\_obs = FALSE,   
 max\_ages = 24,   
 RESOURCE\_ini = 250, # start N   
 culling = FALSE,  
 land\_ownership = TRUE,  
 age\_repr = 4,  
 land\_dim\_1 = DIM\_1,  
 land\_dim\_2 = DIM\_2,  
 manage\_target = 2500,  
 get\_res = 'Full');  
old\_obs <- sim\_old[["observation\_array"]];  
  
# Add crop types on layer 1 of the landscape with the crop\_type(2) function  
sim\_old[["LAND"]] <- crop\_type2(land = sim\_old[["LAND"]]);  
  
# Initialise the results matrix for the current iteration; Currently, 3 variables are recorded as output  
# Double-brackets: technically a bit more secure. It works if you edit columns of the resource\_array directly.  
sim\_sum\_1 <- matrix(data = NA, nrow = TMAX, ncol = 3);  
colnames(sim\_sum\_1) <- c("Time", "Pop\_size", "Pop\_est")  
moving\_month <- 0;  
  
for(time\_step in 1:TMAX){  
 sim\_new <- gmse\_apply(old\_list = sim\_old, get\_res = 'Full',  
 res\_move\_type = moving\_month);  
 sim\_sum\_1[time\_step, 1] <- time\_step;  
 sim\_sum\_1[time\_step, 2] <- sim\_new$basic\_output$resource\_results[1];  
 sim\_sum\_1[time\_step, 3] <- sim\_new$basic\_output$observation\_results[1];  
   
 next\_time <- time\_step + 1; # What happens next time step?  
   
 if(next\_time %in% JAN){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.976083968; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% FEB){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.94824969; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% MAR){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.9023544; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% APR){  
 moving\_month <- 1; # Movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.811569975; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% MAY){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.805694059; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% JUN){  
 moving\_month <- 1; # Movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.794513672; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 1.19; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% JUL){  
 moving\_month <- 1; # Movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.767700722; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 1.19; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% AUG){  
 moving\_month <- 1; # Movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.842305069; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res)  
 }  
 if(next\_time %in% SEP){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.877005333; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
 temp\_res <- sim\_new[["resource\_array"]];  
 sim\_new[["resource\_array"]] <- cell\_K(res = temp\_res, DIM\_1 = DIM\_1,   
 DIM\_2 = DIM\_2);  
  
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
   
 if(next\_time %in% OCT){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.891729601; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
   
   
 }  
 if(next\_time %in% NOV){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.939895976; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 if(next\_time %in% DEC){  
 moving\_month <- 0; # No movement   
 sim\_new[["resource\_array"]][, 9] <- 1 - 0.956833861; # Death probability  
 sim\_new[["resource\_array"]][, 10] <- 0; # Birth probability  
 sim\_new[["observation\_array"]] <- old\_obs; # Use old observations  
   
 temp\_res <- sim\_new[["resource\_array"]];  
 temp\_land <- sim\_new[["LAND"]];  
 sim\_new[["resource\_array"]] <- crop\_hamster(land = temp\_land,   
 res = temp\_res);  
 }  
 sim\_old <- sim\_new; # This should always go at the end  
   
 #print(sim\_sum\_1[time\_step,]); #Activate to see the simulation progress  
}  
  
}, error = function(e) { #tryCatch: if error occurs because N = 0  
 cat("Error occurred in iteration", rep, ":", conditionMessage(e), "\n")  
}) #end of tryCatch  
  
sim\_sum\_1 <- cbind(rep, sim\_sum\_1); # Add iteration number as a column  
print(sim\_sum\_1);   
results[[rep]] <- sim\_sum\_1  
  
###################################### Extra function to save landscape layout for each simulation  
# Get the current timestamp  
timestamp <- format(Sys.time(), "%Y%m%d-%H%M%S")  
  
# Crop type plot  
# Adjust plot margins to make space for the legend  
output\_folder<- "C:/Users/imket/Documents/UAntwerpen/Master thesis/Figures/Landscape simulations/Simulation 8"  
filename\_crop <- file.path(output\_folder, paste0(timestamp, ".png"))  
png(filename\_crop, width = 1000, height = 800)  
par(mar = c(5, 4, 4, 8) + 0.1)  
image(z = sim\_old$LAND[,,1], col = c("#FF6961", "#EBD657", "#92C37F"), xaxt = "n", yaxt = "n",   
 main = "Crop Type Distribution", bty = "n", cex.main = 1.5, cex.lab = 1.2)  
legend("right", inset = c(-0.13, 0), legend = c("Crop Type 1", "Crop Type 2", "Crop Type 3"),   
 fill = c("#FF6961", "#EBD657", "#92C37F"), xpd = TRUE, cex = 1) # to show crop type distribution in the landscape  
dev.off()  
#####################################  
  
}  
  
combined\_results <- do.call(rbind, results)  
combined\_results[is.na(combined\_results)] <- 0  
  
#now timestep is also recorded as a 0 from the moment extinction occurred, we want time step to be recorded correctly  
  
combined\_results <- as.data.frame(combined\_results)  
  
# Loop through each repetition  
for (rep\_id in unique(combined\_results$rep)) {  
 # Subset data for the current repetition  
 rep\_data <- combined\_results[combined\_results$rep == rep\_id, ]  
   
 # Loop through each row in the current repetition's data  
 for (i in 2:nrow(rep\_data)) {  
 # If the current time step is 0, replace it with the previous time step + 1  
 if (rep\_data$Time[i] == 0) {  
 rep\_data$Time[i] <- rep\_data$Time[i - 1] + 1  
 }  
 }  
   
 # Replace the data for the current repetition in the original data frame  
 combined\_results[combined\_results$rep == rep\_id, ] <- rep\_data  
}  
  
file\_path <- "C:/Users/imket/Documents/GitHub/European\_hamster\_GMSE/Output/Simulation8\_100it\_250startN.csv" #Give the simulation an appropriate name  
write.csv(combined\_results, file = file\_path, row.names = FALSE)

# Crop type plot  
# Adjust plot margins to make space for the legend  
par(mar = c(2, 3, 2, 8) + 0.2)  
image(z = sim\_old$LAND[,,1], col = c("#FF6961", "#F8D66D", "#8CD47E"), xaxt = "n", yaxt = "n", main = "Crop type distribution", bty = "n")  
legend("right", inset = c(-0.30, 0), legend = c("Crop Type 1", "Crop Type 2", "Crop Type 3"),   
fill = c("#FF6961", "#F8D66D", "#8CD47E"), xpd = TRUE, cex = 1) #to show crop type distribution in the landscape  
  
# Stakeholders plot   
# Define the range of stakeholder numbers  
stholders <- 2:11 #adjust based on number of stakeholders   
  
# Generate a color palette with enough distinct colors for each stakeholder  
colours <- rainbow(length(stholders), s = 0.5)  
image(z = sim\_old$LAND[,,3], col = colours, xaxt = "n", yaxt = "n", main = "Stakeholder land distribution", bty = "n")