Tic-Tac-Toe: Individual-based spatial consumer-resource disease transmission model for predicting parasite loading on nutrient cycling in ecosystems

Matt Malishev, Emory University, USA
J. Trevor Vannatta, Purdue University, USA
Amanda Koltz, Washington University in St. Louis, USA
Rachel Penczkowski, Washington University in St. Louis, USA
Sharon Deem, Institute for Conservation Medicine, St. Louis Zoo, USA
Vanessa Ezenwa, University of Georgia, USA
Zoe Johnson, Mississippi State University, USA
Aimee Classen, University of Vermont, USA
Maris Brenn-White, St. Louis Zoo, USA
David Civitello, Emory University, USA

Contents

Date: 2019-01-25
R version: 3.5.0
*Corresponding author: matthew.malishev@gmail.com
This document can be found at https://github.com/darwinanddavis/tictactoe

Overview

Develop the consumer-resource disease transmission model of parasite loading on nutrient cycling in ecosystems as a spatial individual-based model.

To forecast how resource biomass uptake and release by infected and non-infected host populations varies under a disease mosaic landscape driven by feedback between modes and rates of disease transmission and costs of parasite occurrence and nutrient deposit in space and time.

The model applies the nutrient-plant-susceptible-infectious (NPSI) model in a spatial landscape of resources, host populations, and disease vector populations.

The rundown

The model explores the transmission probability of diseases among host populations in a simulated patchwork landscape of resource and disease patches. The two main entities are host individuals (agents) and patches. Hosts are mobile units that move throughout the model landscape using user-defined movement/patch occurrence rules and aim to consume the resource patches that provide them with energy to fuel their growth/death rates. Individuals belong to host populations that are either susceptible to infection risk (S) or infected with a disease (I).

The probability of a susceptible host becoming infected is determined via two pathways:

- Encounter food and/or ground patches in the landscape that contain parasites, such as diseased food or infected waste (faeces or carcasses of infected hosts).
- Encounter infected hosts on a shared food and/or ground patch.

Infected hosts can transmit parasites to susceptible hosts. Infected hosts cannot be infected further, but can shed their parasites, after which they reset their infection probability and become susceptible again.

Patches in the landscape are either ground, resources, or infected patches (ground or resources). Resource patch growth rates depend on nutrient supply in a patch. Bare ground patches can become resource patches when their nutrient load is sufficient to grow resources. Resource patches are consumed if 1) the patch is occupied by individual/multiple hosts and 2) the host requires energy.

All rates of nutrient supply, resource growth, and host population densities are determined by the state variable ODEs (defined below).

From individual to population

The model can be as simple or as complex as we want it to be. The best way to build the model is to keep our main research questions at the forefront as we modify the model assumptions. This will help us confine the aims of the model and what we feed into it, which is limited by available data.

The model uses individual hosts as units and patches as cells that contain information on how the individual will update its current state at each time step. For example, a host that begins the simulation at time step 0 as susceptible will encounter food patches at different time steps, which update its energetic state. If this host then encounters an infected host at time step 5, the state of this newly infected host will change to infected. This newly infected host then updates its current energetic state at time step 6 to reflect the consequences of its body condition being infected.

The nutrient and resource load of patches also varies per time step as patches are updated with new nutrients, consumed by hosts, or infected by infected products or carcasses.

While the units are the individual hosts, the patterns emerging from the simulation are at the host population level. This keeps the transmission dynamics within a susceptible/infected population framework that corresponds to the output of biomass back into the landscape. The rates of transmission and resource

growth are at faster time scales than host birth/death rates. Therefore, varying these rates and the values that feed these rates will determine what the model simulations produce throughout the different time and space scales.

Methods

Model description

State variables (units = biomass)
N = nutrients in the landscape (biomass)
P = food in the landscape (plant biomass)
S = susceptible ungulate host population
I = infected ungulate host population

Parameters
r = intrinsic growth rate of plants
K = carrying capacity of plants
a = rate of nutrient addition
l = nutrient loss rate
fp = rate of plant nutrient uptake
β = transmission rate
es = assimilation efficiency of susceptible hosts
ei = assimilation efficiency of infected hosts
fs = feeding rate of susceptible hosts
fi = feeding rate of infected hosts
d = background host death rate
v = mortality rate from infection
ws = rate of waste production from susceptible hosts
wi = rate of waste production from infected hosts

State variable ODEs

Nutrient growth (biomass)

$$\frac{dN}{dt} = a - lN - fpNP + (d + ws)S + (d + v + wi)I$$

Food growth (biomass)

$$\frac{dP}{dt} = fpNrP(1 - (P/K)) - P(fsS + fiI)$$

Susceptible host density

$$\frac{dS}{dt} = P(esfsS + eifiI) - \beta S - (d + ws)S$$

Infected host density

$$\frac{dI}{dt} = \beta S - (d + v + wi)I$$

The interplay between the fast (e.g. feeding, disease transmission) and slow (e.g. nutrient cycling, birth/death) rate dynamics at different time and space scales generates emergent patterns for the different state variable and parameter values we use to define the model starting conditions and inputs.

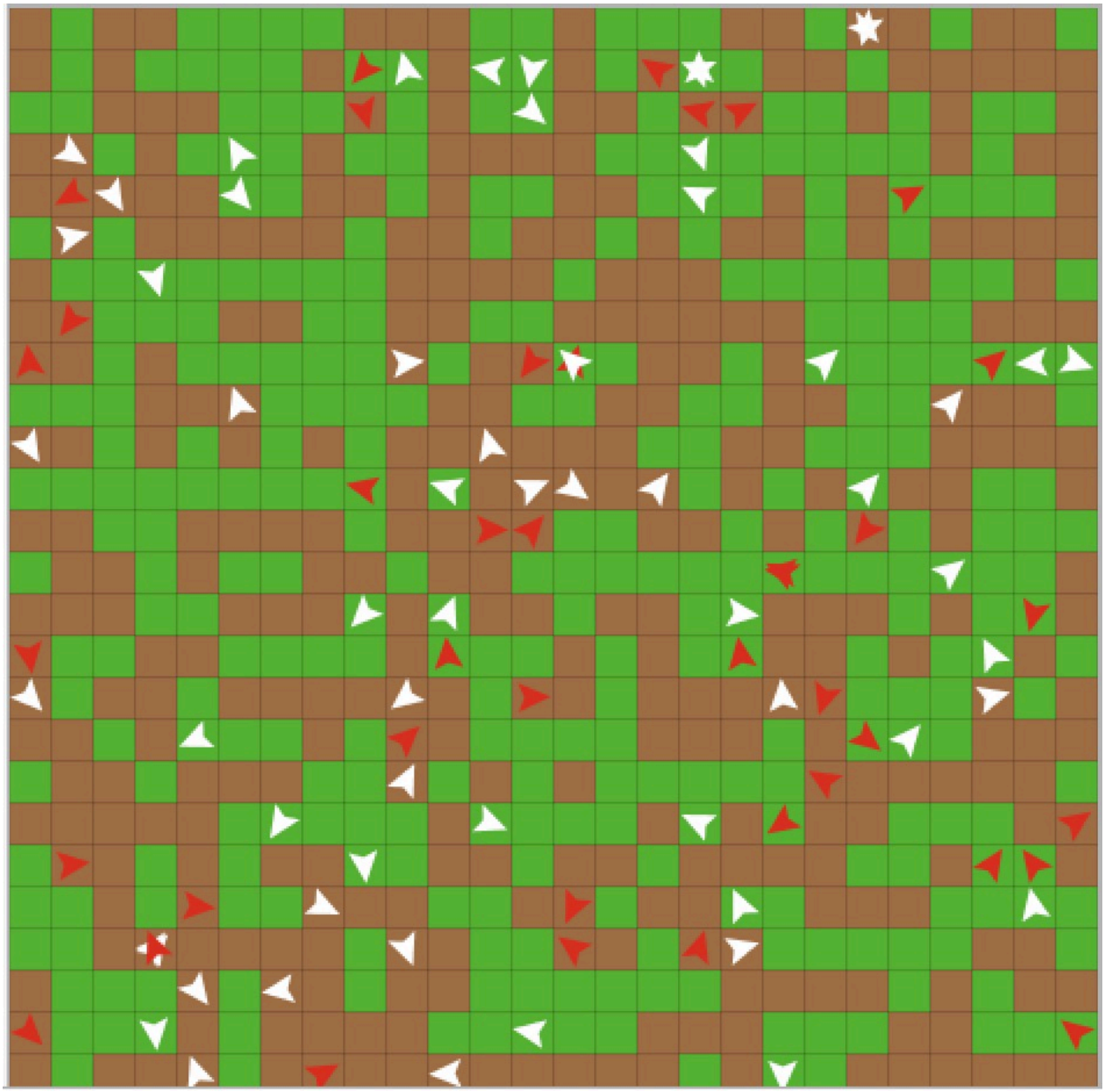

Figure 1: Example of what the model landscape might look like with food patches (green), infected patches (brown), and host individuals (white = S, red = I).

Questions to answer

Some examples of questions we could answer:

- What density of infected patches in the landscape increases transmission and mortality rates?
- How does host density per patch drive disease transmission rates?
- What density of infected resources drives direct mortality (host density per infected patch) versus indirect mortality (horizontal disease transmission from newly born infected hosts)?
- How does persistence of infected host density shape infected patch arrangement in the landscape and ultimately exposure risk to susceptible hosts?
- What density of infected hosts suppresses nutrient stocks to levels below unsustainable food growth?
- Etc

Extensions of the model

The model landscape can also include free-ranging, mobile disease vectors, i.e. mosquitoes and ticks, that have their own basic movement rules. This can follow three modes:

- Opportunistic, where vectors follow correlated random walk in the landscape and host-vector encounter rates are probabilistic based on vector and susceptible host density per patch.
- Recurring, where vector density in the landscape peaks according to regular rollout events throughout the simulation. This aims to follow vector occurrence patterns that correlate with predictable natural phenomena i.e. seasonal fluctuations in humidity.
- Episodic, where the landscape is 'flooded' with vectors at given times throughout the simulation. This aims to replicate sudden disease outbreaks in a given spatial area that may correspond to climate, human-induced, and/or epidemic events.

Appendix

Model code (from nutrient-plant-susceptible-infectious (NPSI) model)

```
#########################################################################################
################################## User inputs for model ################################
#########################################################################################

# set your working directory
# E.g "/Users/malishev/dope_models/my_dope_model/"
wd <- "paste the path to where you saved the model here (with these quotes)"
setwd(wd)

# set parameter ranges (min 0, max 1)
beta_access <- 0.1 # choose your beta value you want to plot at the end
death_access <- 0.9 # choose your death value you want to plot at the end
colvv <- "orange" # choose your plot line colour

## initial conditions
N <- 200 # size of nutrient biomass in env
P <- 200 # initial products in env
S <- 20 # num of susceptible hosts
In <- 2 # num of infected hosts
```

```r
years <- 100 # number of years to run simulation
time.out <- 0.01 # simulation time step (0.01 = 1 year if years = 100)
```

```r
#########################################################################################################
#################################### Setup simulation model ##############################
#########################################################################################################
# set param space
beta_pars <- seq(0.1,1,0.1) # transmission rate in model
death_pars <- seq(0.1,1,0.1) # death rate in model

# desired outputs
out <- list()
out_master <- list() # NPSI output
out_tibble <- tibble()
outplot <- list()
param_space <- list(beta_pars,death_pars) # summed parameter space

# create empty list
out_master <- rep(
  list(structure(list(
    pars = numeric(),
    outs = list()
    ),
    .Names = c("Parameter", "Output")))
    ,prod(as.numeric(summary(param_space)[,1]))
  )
sc <- 1 # timer in simulation model

#########################################################################
# create simulation model  ############################

# to set pars as individual beta and death values
npsi_func <- function(){ # start npsi_func

  # ------- start simulation # -------
  for(beta in beta_pars){ # pass through beta values
    for(death in death_pars){ # pass through death values
      parameters<-c(r=0.2, K=100, a=500, l=5, fp=0.5, beta=beta,
                    es=0.1, ei=0.05, fs=0.2, fi=0.1,
                    d=death, v=0.1, ws=0.05, wi=0.09)

      state<-c(N=N, P=P, S=S, I=In) # set initial conditions

      NPSI<-function(t, state, parameters) {
        with(as.list(c(state, parameters)),{

          dN.dt <- a - l*N - fp*N*P + (d+ws)*S + (d+v+wi)*I  # nutrients in env
          dP.dt <- fp*N*r*P*(1-(P/K)) - P*(fs*S*+fi*I) # plants produced
          dS.dt <- P*(es*fs*S + ei*fi*I) - beta*S - (d+ws)*S # susceptible hosts
          dI.dt <- beta*S - (d+v+wi)*I # infected hosts

          list(c(dN.dt, dP.dt, dS.dt, dI.dt)) # compile outputs
        })
      } # end npsi function
```

```
      # -------  global output # -------
      times <- seq(0, years, by=time.out) # set time horizon for simulation (years)
      out <- ode(y=state, times=times, func=NPSI, parms=parameters) # run sim
      out <- data.frame(out)
      # save outputs
      # out_master[[length(out_master) + 1]] <- out # working with out_master <- list()
      out_master[[sc]]$Output <- out # save output for each run
      out_master[[sc]]$Parameter[1] <- beta # save beta for each run
      out_master[[sc]]$Parameter[2] <- death # save death for each run
      sc <- sc + 1
    } # end death pars
  } # end beta pars

  # -------  clean output # -------
  # save simulation model to global vector (tibble)
  out_tibble <- tibble(
    params = map(out_master, "Parameter"),
    outs = map(out_master, "Output")
  ) %>%
    mutate(
      beta = map(params, 1),
      death = map(params, 2)
    ) %>%
    select(beta, death, outs)

  # ------- plotting ----------
  # start save plot to local dir
  pdf(paste0(getwd(),"/npsi_model_plot.pdf"),onefile=T,width=10,height=8,paper="a4r")
  outplot <- filter(out_tibble, death == death_access & beta == beta_access)
  outplot <- outplot$outs ; outplot <- as.data.frame(outplot) # clean output
  outplot$"Total host population" <- outplot[,"S"] + outplot[,"I"] # sum hosts
  # plot results
  layout(matrix(c(1,2,3,4,5,5), 2, 3, byrow = TRUE)) # set plot window
  colnames(outplot) <- c("Time",
                         "Nutrient biomass",
                         "Product biomass",
                         "Susceptible host pop",
                         "Infected host pop",
                         "Total host population")
  for (name in names(outplot)[c(3:5,2,6)]){ # start plot
    plot(outplot[,1],outplot[,name],type="l",las=1,bty="n",
         xlab="Time (years)",ylab=name,col=colvv,
         ylim=c(0,round_any(max(outplot[,name]),10,ceiling))
    )
  } # end plot
  # add mean plot
  dev.off() # save output to dir
  cat(paste0("\n\n\nPlot is saved in \n",getwd(), "\nas npsi_model_plot.pdf\n\n\n"))
  return(out_tibble)
} # ------- end npsi_func

### run model function
out_tibble <- npsi_func()
```

```
#########################################################################################
#################################### end simulation model ###############################
#########################################################################################

################################### plot results manually  ##############################

# set parameter ranges (min 0, max 1)
beta_access <- 0.1 # choose your beta value you want to plot at the end
death_access <- 0.9 # choose your death value you want to plot at the end
colvv <- "orange" # choose your plot line colour

# then run this part to plot in your live R session

outplot <- filter(out_tibble, death == death_access & beta == beta_access)
outplot <- outplot$outs ; outplot <- as.data.frame(outplot) # clean output
outplot$"Total host population" <- outplot[,"S"] + outplot[,"I"] # sum hosts

layout(matrix(c(1,2,3,4,5,5), 2, 3, byrow = TRUE)) # set plot window
colnames(outplot) <- c("Time",
                       "Nutrient biomass",
                       "Product biomass",
                       "Host population size \n(susceptible)",
                       "Host population size \n(infected)",
                       "Total host population")
for (name in names(outplot)[c(3:5,2,6)]){ # start plot
  plot(outplot[,1],outplot[,name],type="l",las=1,bty="n",
       xlab="Time (years)",ylab=name,col=colvv,
       ylim=c(0,round_any(max(outplot[,name]),10,ceiling))
  )
} # end plot
```