Model debug

Chris Hoover

October 11, 2018

Some model runs encounter an error when estimating the mating probability that causes an error in the integrate function. Need to look into this more as just skipping over parameter sets that cause this error may intriduce bias

First need to reproduce the error

```
#Load all necessary functions, models, and parameter sets
source('../../R/Prawn_aquaculture/prawn_aquaculture_mod.R')
source('../../R/Prawn_aquaculture/prawn_aquaculture_sim.R')
source("../../R/Epi_Model/snail_epi_mod_no_diag_immigration.R")
source("../../R/Combined_Model/epi_no_diag_prawn_immigration_mod.R")
source("../../R/Combined_Model/epi_no_diag_prawn_sim.R")
source("../../Data/snail_epi_parameters.R")
source("../../Data/combined_parameters.R")
source("../../Data/aquaculture_parameters.R")
source("../../Data/Ranjeet_Kurup_06_data.R")

  eta.lm = lm(marketable ~ dens, data = rk06)

source("../../R/Sensitivity_Analysis/sensitivity_prep.R")

par.all = c(par.aqua, par.snails.imm, par.epi_prawn)

#Load some parameters from the prawn aquaculture simulations needed here
qwraps2::lazyload_cache_labels("Opt_stock",
                               path = "../../Analysis/Prawn_Aquaculture_Profit_Maximization_cache/latex/")
```

```
## Lazyloading ../../Analysis/Prawn_Aquaculture_Profit_Maximization_cache/latex/Opt_stock_eebfb30b361fdebc74ef
```

```
#Run to equilibrium to get eqbm values of state variables
allvh.eqbm.run.imm0 = as.data.frame(ode(nstart.sn,
                                        t.sn,
                                        snail_epi_allvh_imm,
                                        par.snails.imm))

  allvh.eqbm.imm0 = allvh.eqbm.run.imm0[dim(allvh.eqbm.run.imm0)[1],]

#Add immigration parameters based on immigration-free eqbm, add immigration parameter
  par.snails.imm["xi"] <- par.all["xi"] <-  0.2/365  # 20% per year as in Head et al snail migration paper
  par.snails.imm["siteS1"] <- par.all["siteS1"] <- allvh.eqbm.imm0$S1
  par.snails.imm["siteS2"] <- par.all["siteS2"] <- allvh.eqbm.imm0$S2
  par.snails.imm["siteS3"] <- par.all["siteS3"] <- allvh.eqbm.imm0$S3
  par.snails.imm["siteE1"] <- par.all["siteE1"] <- allvh.eqbm.imm0$E1
  par.snails.imm["siteE2"] <- par.all["siteE2"] <- allvh.eqbm.imm0$E2
  par.snails.imm["siteE3"] <- par.all["siteE3"] <- allvh.eqbm.imm0$E3
  par.snails.imm["siteI1"] <- par.all["siteI1"] <- allvh.eqbm.imm0$I1
  par.snails.imm["siteI2"] <- par.all["siteI2"] <- allvh.eqbm.imm0$I2
  par.snails.imm["siteI3"] <- par.all["siteI3"] <- allvh.eqbm.imm0$I3

#Rerun to eqbm with immigration
allvh.eqbm.run.imm = as.data.frame(ode(setNames(as.numeric(allvh.eqbm.imm0[-1]), names(allvh.eqbm.imm0)[-1]),
                                       t.sn,
                                       snail_epi_allvh_imm,
                                       par.snails.imm))
```

```r
#Save for eqbm starting values in further simulations
  allvh.eqbm.imm = allvh.eqbm.run.imm %>% filter(time == max(time)) %>% select(S1:Wu) %>% unlist()


## Set initial values and parameters ###########
t.all = c(1:(365*years*2))+365

#Starting values from model equilibrium
nstart = allvh.eqbm.imm

#Volenhovenii stocking and harvesting events ###########
#Create data frame representing stocking and harvesting events for M. vollenhovenii at optimal management
  harvest.t.vol = opt.vol$time           #Optimal harvest time from volenhovenii optimization
  n.harvest.vol = floor((years*365)/harvest.t.vol)
  stocks.vol = data.frame(var = rep(c('P', 'L'), n.harvest.vol+1),
                          time = rep(366 + harvest.t.vol*c(0:n.harvest.vol), each = 2),
                          value = rep(c(opt.vol$P_nought, opt.vol$L_nought), n.harvest.vol+1),
                          method = rep('rep', (n.harvest.vol+1)*2))

    vol.harvests = stocks.vol[seq(1, nrow(stocks.vol), 2), 2]

  par.all.vol = par.all
    par.all.vol['k'] <- k.vol

#Rosenbergii stocking and harvesting events ###########
#Create dataframe represnting stocking and harvesting of M. rosenbergii prawns
  harvest.t.ros = opt.ros$time
  n.harvest.ros = floor((years*365)/harvest.t.ros)
  stocks.ros = data.frame(var = rep(c('P', 'L'), n.harvest.ros+1),
                          time = rep(366 + harvest.t.ros*c(0:n.harvest.ros), each = 2),
                          value = rep(c(opt.ros$P_nought, opt.ros$L_nought), n.harvest.ros+1),
                          method = rep('rep', (n.harvest.ros+1)*2))

    ros.harvests = stocks.ros[seq(1, nrow(stocks.ros), 2), 2]

  par.all.ros = par.all
    par.all.ros['k'] = k.ros  # alternate value for M. rosenbergii, from Sampaio & Valenti 1996: 0.0104333333

#mda events #######
  mdas = data.frame(var = rep('Wt', years+1),
                    time = 365*c(0:years+1)+1,
                    value = rep((1-eff), years+1),
                    method = rep('multiply', years+1))

#Combined MDA and vollenhovenii stocking events #######
  mda.vol = rbind(mdas, stocks.vol)
  mda.vol = mda.vol[order(mda.vol$time),]

#Combined MDA and rosenbergii stocking events #######
  mda.ros = rbind(mdas, stocks.ros)
  mda.ros = mda.ros[order(mda.ros$time),]

#allow for one year run in to display epi model at eqbm ###########
t.yr1 = c(0:365)

yr1 = as.data.frame(ode(nstart,
                        t.yr1,
                        snail_epi_allvh_imm,
                        par.snails.imm)) %>%
```

```
  mutate(P = 0, #Add prawn variable to integrate with prawn stocking sims
         L = 0, #Add prawn variable to integrate with prawn stocking sims
         W = cvrg*Wt + (1-cvrg)*Wu,
         prev = pnbinom(2, size = par.snails.imm['phi'], mu = W, lower.tail = FALSE),
         S.t = (S1 + S2 + S3) / area,  # density susceptible snails
         E.t = (E1 + E2 + E3) / area,  # density exposed snails
         I.t = (I1 + I2 + I3 ) / area, # density infected snails
         N.t = (S.t + E.t + I.t), # total density snails
         t.1 = (S1 + E1 + I1) / area, # density snails of size class 1
         t.2 = (S2 + E2 + I2) / area, # density snails of size class 2
         t.3 = (S3 + E3 + I3) / area) # density snails of size class 3

# Prep parameter sets and first year for sims with no migration #########
#Simulations with migration to/from external sources absent
  par.all.vol.nomig  <- par.all.vol
  par.all.ros.nomig  <- par.all.ros
  par.snails.nomig <- par.snails.imm

  par.all.vol.nomig["xi"] <- 0
  par.all.ros.nomig["xi"] <- 0
  par.snails.nomig["xi"] <- 0

#allow for one year run in to display epi model at eqbm
yr1.nomig = as.data.frame(ode(nstart,
                              t.yr1,
                              snail_epi_allvh_imm,
                              par.snails.nomig))  %>%
  mutate(P = 0, #Add prawn variable to integrate with prawn stocking sims
         L = 0, #Add prawn variable to integrate with prawn stocking sims
         W = cvrg*Wt + (1-cvrg)*Wu,
         prev = pnbinom(2, size = par.snails.nomig['phi'], mu = W, lower.tail = FALSE),
         S.t = (S1 + S2 + S3) / area,  # density susceptible snails
         E.t = (E1 + E2 + E3) / area,  # density exposed snails
         I.t = (I1 + I2 + I3 ) / area, # density infected snails
         N.t = (S.t + E.t + I.t), # total density snails
         t.1 = (S1 + E1 + I1) / area, # density snails of size class 1
         t.2 = (S2 + E2 + I2) / area, # density snails of size class 2
         t.3 = (S3 + E3 + I3) / area) # density snails of size class 3
```

```
qwraps2::lazyload_cache_labels("compare_interventions", path = "../../Analysis/Epi_simulations_cache/latex/")
```

```
## Lazyloading ../../Analysis/Epi_simulations_cache/latex/compare_interventions_4d3997d4aa19142adf121f22c68d55
```

```
# This parameter set reproduces the error, commented out to the document knits
  # compare_interventions(unlist(lhcpars[2,]), years = years)

pars <- unlist(lhcpars[2,])
```

Error is reproduced with the above parameter set, let's dive into the functions a bit more to see what's going on. Redefine the epi model so that it prints W at each time step, then re run the same function

```
#mating function
fx<-function(x, mean.worm, clump){
  alpha<-(mean.worm)/(clump+mean.worm)
  (1-cos(x)) / ( (1+(alpha*cos(x)))^(1+clump) )
}

phi_Wk<-function(W, phi){
```

```
  alpha<-W/(W+phi)
  1-( (1-alpha)^(phi+1) * (integrate(fx, 0, 2*pi, W, phi, stop.on.error = F)$value) /(2*pi)  )
}

#Version of function that prints  state variables at each time step ######
snail_prawn_model_imm = function(t, n, parameters) {
  with(as.list(parameters),{

    S1=n[1]
    S2=n[2]
    S3=n[3]
    E1=n[4]
    E2=n[5]
    E3=n[6]
    I1=n[7]
    I2=n[8]
    I3=n[9]
    Wt=n[10]
    Wu=n[11]
    P=n[12]
    L=n[13]

    N1 = S1+E1+I1     # Total snails of size class 1
    N2 = S2+E2+I2  # Total snails of size class 2
    N3 = S3+E3+I3  # Total snails of size class 3
    N = N1+N2+N3   # Total number of snails
    W = cvrg*Wt + (1-cvrg)*Wu  #Mean worm burden weighted between treated and untreated populations

    #  print(c(t, N1, N2, N3, P, L, W, Wt, Wu))

    # Miracidial density per person as a function of mean worm burden (W) and miracidial shedding rate (m)
    mate = phi_Wk(W = W, phi = phi)  #Mating probability
    M = m*0.5*W*mate

    # Mean and total prawn biomass, converting from length (cm) to weight (g)
    Bm.p = 10^a.p*(L/10)^b.p
    Bm.t = P*Bm.p

    # Mean snail biomass in each size class, converting from length (cm) to weight (g)
    Bm.n1 = a.s*0.4^b.s  # 4mm size class
    Bm.n2 = a.s*0.8^b.s  # 8mm size class
    Bm.n3 = a.s*1.2^b.s  # 12mm size class

    # Prawn-to-snail mass ratios for each size class
    Bm.r1 = Bm.p / Bm.n1
    Bm.r2 = Bm.p / Bm.n2
    Bm.r3 = Bm.p / Bm.n3

    # Attack rates for each size class as a function of biomass ratio (3 is biomass ratio below which refuge e
    alpha1 = ifelse(Bm.r1 > 3, ar.slp*log(Bm.r1), 0)  #if biomass ratio <3, size refuge exists (from Sokolow 2
    alpha2 = ifelse(Bm.r2 > 3, ar.slp*log(Bm.r2), 0)  #if biomass ratio <3, size refuge exists (from Sokolow 2
    alpha3 = ifelse(Bm.r3 > 3, ar.slp*log(Bm.r3), 0)  #if biomass ratio <3, size refuge exists (from Sokolow 2

    #alpha1 = ifelse(-log(3) + ar*log(Bm.r1) < 0, -log(3) + ar*log(Bm.r1), 0)

    # Adjusted attack rates, accounting for area of interest and limiting factors in the wild
    alpha_star1 = alpha1/eps
    alpha_star2 = alpha2/eps
    alpha_star3 = alpha3/eps
```

```
# Handling times for each size class as a function of biomass ratio
handle1 = 1/(th*Bm.r1)
handle2 = 1/(th*Bm.r2)
handle3 = 1/(th*Bm.r3)

# Functional responses for each size/infection class
psiS1 = (alpha_star1*(S1/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiS2 = (alpha_star2*(S2/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiS3 = (alpha_star3*(S3/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiE1 = (alpha_star1*(E1/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiE2 = (alpha_star2*(E2/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiE3 = (alpha_star3*(E3/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
```

```
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiI1 = (alpha_star1*(I1/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiI2 = (alpha_star2*(I2/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

psiI3 = (alpha_star3*(I3/A)^nfr) / (1 + sum(alpha_star1*handle1*(S1/A),
                                        alpha_star2*handle2*(S2/A),
                                        alpha_star3*handle3*(S3/A),
                                        alpha_star1*handle1*(E1/A),
                                        alpha_star2*handle2*(E2/A),
                                        alpha_star3*handle3*(E3/A),
                                        alpha_star1*handle1*(I1/A),
                                        alpha_star2*handle2*(I2/A),
                                        alpha_star3*handle3*(I3/A))^nfr)

#print(c(psiS1, psiS2, psiS3, psiE1, psiE2, psiE3, psiI1, psiI2, psiI3)*P)

## Model equations:
dS1dt = xi*siteS1 + f*(1-N/Kn)*(S2+S3+z*(E2+E3)) - muN1*S1 - psiS1*P - g1*S1 - (beta)*M*H*S1 - xi*S1

dS2dt = xi*siteS2 + g1*S1 - muN2*S2 - psiS2*P - g2*S2 - (beta)*M*H*S2 - xi*S2

dS3dt = xi*siteS3 + g2*S2 - muN3*S3 - psiS3*P - (beta)*M*H*S3 - xi*S3

dE1dt = xi*siteE1 + (beta)*M*H*S1 - muN1*E1 - psiE1*P - sigma*E1 - g1*E1 - xi*E1

dE2dt = xi*siteE2 + (beta)*M*H*S2 + g1*E1 - muN2*E2 - psiE2*P - sigma*E2 - g2*E2 - xi*E2

dE3dt = xi*siteE3 + (beta)*M*H*S3 + g2*E2 - muN3*E3 - psiE3*P - sigma*E3 - xi*E3

dI1dt = xi*siteI1 + sigma*E1 - (muN1+muI)*I1 - psiI1*P - g1*I1 - xi*I1

dI2dt = xi*siteI2 + sigma*E2 + g1*I1 - (muN2+muI)*I2 - psiI2*P - g2*I2 - xi*I2

dI3dt = xi*siteI3 + sigma*E3 + g2*I2 - (muN3+muI)*I3 - psiI3*P - xi*I3

dWtdt = lambda*I1 + theta1*lambda*I2 + theta2*lambda*I3 - (muW + muH)*Wt
```

```
    dWudt = lambda*I1 + theta1*lambda*I2 + theta2*lambda*I3 - (muW + muH)*Wu

    # Prawn abundance
    dPdt = -P*(muP*Bm.p^d + om*Bm.t)

    # Mean prawn length (mm)
    dLdt = k/(1+gam*Bm.t)*(linf - L)


    return(list(c(dS1dt, dS2dt, dS3dt, dE1dt, dE2dt, dE3dt, dI1dt, dI2dt, dI3dt, dWtdt, dWudt, dPdt, dLdt)))
  })
}

#First part of `compare interventions function` ######
   sim_prawns <- as.data.frame(ode(prawn_start,
                                   c(1:(365*2)),
                                   prawn_biomass,
                                   pars)) %>%
      mutate(p_mass = 10^(pars["a.p"]+pars["b.p"]*log10(L/10)),
             total_mass = (P*p_mass) /1000,
             harvest_mass = total_mass * as.numeric(pars["eta"]),
      #If average prawn mass is less than 30 g (marketable size) don't estimate monetary outcomes
             revenue = ifelse(p_mass <30, NA, harvest_mass*price),
             profit = ifelse(p_mass <30, NA, revenue*exp(-pars["delta"]*time) - pars["c"]*prawn_start["P"] - p
             n_harvest = ifelse(p_mass <30, NA, floor((years*365)/time)),
             cum_profits = pmap_dbl(list(n = n_harvest,
                                         Pi = profit,
                                         delta = delta,
                                         Time = time), get_cum_profits))

    opt_harvest <- sim_prawns %>% filter(cum_profits == max(cum_profits, na.rm = TRUE))

# Generate data frame of stocking/harvesting events
  harvest_time <- opt_harvest$time
  n.harvest = opt_harvest$n_harvest
  stocks = data.frame(var = rep(c('P', 'L'), n.harvest),
                          time = rep(harvest_time*c(0:(n.harvest-1))+1, each = 2),
                          value = rep(c(opt.ros$P_nought, opt.ros$L_nought), n.harvest),
                          method = rep('rep', (n.harvest)*2))

    harvests = stocks[seq(1, nrow(stocks), 2), 2]

#mda events #######
  mdas = data.frame(var = rep('Wt', years),
                    time = 365*c(0:(years-1))+1,
                    value = rep((1-eff), years),
                    method = rep('multiply', years))

#Combined prawn stocking and mda events
  mda.prawn = rbind(mdas, stocks)
  mda.prawn = mda.prawn[order(mda.prawn$time),]

# Run epi model to equilibrium
  epi_sim <- as.data.frame(ode(nstart,
                               seq(1,365*100,25),
                               snail_epi_allvh_imm,
                               pars))

  epi_eqbm <- epi_sim %>% filter(time == max(time))
```

```
    full_eqbm <- epi_eqbm %>% select(-time) %>% unlist()


#Add in prawn info and migration info to starting conditions/parameter set
  full_eqbm["P"] <- opt.ros$P_nought
  full_eqbm["L"] <- opt.ros$L_nought

  pars["siteS1"] <- epi_eqbm %>% pull(S1)
  pars["siteE1"] <- epi_eqbm %>% pull(E1)
  pars["siteI1"] <- epi_eqbm %>% pull(I1)
  pars["siteS2"] <- epi_eqbm %>% pull(S2)
  pars["siteE2"] <- epi_eqbm %>% pull(E2)
  pars["siteI2"] <- epi_eqbm %>% pull(I2)
  pars["siteS3"] <- epi_eqbm %>% pull(S3)
  pars["siteE3"] <- epi_eqbm %>% pull(E3)
  pars["siteI3"] <- epi_eqbm %>% pull(I3)
```

```
# Part of `compare interventions` script that was throwing an error
  #Simulate combined intervention under optimal management and annual MDA for n years
  cmbnd_sim <- as.data.frame(ode(full_eqbm,
                                 c(1:(years*365)),
                                 snail_prawn_model_imm,
                                 pars,
                                 events = list(data = mda.prawn))) %>%
    mutate(W = cvrg*Wt + (1-cvrg)*Wu,
           prev = pnbinom(2, size = pars['phi'], mu = W, lower.tail = FALSE),
           S.t = (S1 + S2 + S3) / area,  # density susceptible snails
           E.t = (E1 + E2 + E3) / area,  # density exposed snails
           I.t = (I1 + I2 + I3 ) / area, # density infected snails
           N.t = (S.t + E.t + I.t), # total density snails
           t.1 = (S1 + E1 + I1) / area, # density snails of size class 1
           t.2 = (S2 + E2 + I2) / area, # density snails of size class 2
           t.3 = (S3 + E3 + I3) / area)
```

Above code not run in order to prevent error out, but found that setting tolerances and switching to method "radau" prevents the error. Fix found in the deSolve vignettes

```
#Simulate combined intervention under optimal management and annual MDA for n years
  cmbnd_sim <- as.data.frame(ode(full_eqbm,
                                 c(1:(years*365)),
                                 snail_prawn_model_imm,
                                 pars,
                                 events = list(data = mda.prawn),
                                 atol = 1e-6, rtol = 1e-6, method = "radau")) %>%
    mutate(W = cvrg*Wt + (1-cvrg)*Wu,
           prev = pnbinom(2, size = pars['phi'], mu = W, lower.tail = FALSE),
           S.t = (S1 + S2 + S3) / area,  # density susceptible snails
           E.t = (E1 + E2 + E3) / area,  # density exposed snails
           I.t = (I1 + I2 + I3 ) / area, # density infected snails
           N.t = (S.t + E.t + I.t), # total density snails
           t.1 = (S1 + E1 + I1) / area, # density snails of size class 1
           t.2 = (S2 + E2 + I2) / area, # density snails of size class 2
           t.3 = (S3 + E3 + I3) / area)
```