```
### KHV MODEL (OC) ===== ===== ===== ===== =====

# last updates: January 23, 2023

library(cowplot)
library(deSolve)
library(ggplot2)
library(graphics)
library(reshape2)
library(tidyverse)

rm(list=ls()) # clear environment


### INITIALIZE ===== ===== ===== ===== =====

## Time =====
yrs = 1 # max = 12
dt = 1
tvals = seq(0,yrs*365,by=dt)
tlen = length(tvals)

## Initial Conditions =====
S0=0
A0=0
I0=0
D0=0
W0=0

inits_sta = c(S=S0, A=A0, I=I0, D=D0, W=W0)

# transversality condition, lambda(T) = 0
inits_lam = c(lambdaS=0, lambdaA=0, lambdaI=0, lambdaD=0, lambdaW=0)


### OC PARAMETERS ===== ===== ===== ===== =====

# max harvesting rate
hmax = 0.01
# profit of harvesting
P = 1
# cost of harvesting
C1 = 10000
# cost of cases
C2 = 0.1
```

```
### INDEPENDENT PARAMETERS ===== ===== ===== ===== =====
mu = 0.00014
betaA = 0.00000088
betaID = 0.0000088
betaW = 0.05
xi = 0.2
rhoA = 0.00001
rhoID = 0.00002
omega = 0.00001
eta = 1/3
p = 0.05
gb = 0.0006
gJ = 0.00009

pars = c(mu=mu, betaA=betaA, betaID=betaID, betaW=betaW, xi=xi,
         rhoA=rhoA, rhoID=rhoID, omega=omega,
         eta=eta, p=p, gb=gb, gJ=gJ,
         C1=C1, C2=C2, P=P, hmax=hmax)


### DEPENDENT PARAMETERS ===== ===== ===== ===== =====

tempfunc = function(t){
  temp = 70 + 30*cos(2*pi*t/365)}
NCfunc = function(temp){
  NC = dnorm(temp,68.9,2.7)}
sigmafunc = function(NC){
  sigma = ((1/5.9)/max(NC))*NC}
alphafunc = function(NC){
  alpha = ((1/5.3)/max(NC))*NC}
gammafunc = function(NC){
  gamma = ((1/21)/max(NC))*(-NC + max(NC))}
mSAfunc = function(t){
  mSA = 100*exp(-0.015*t)}
mIDfunc = function(t){
  mID = 10*exp(-0.015*t)}
bSAfunc = function(t){
  ifelse(t>275+0*365 & t<275+0*365+60 | t>275+1*365 & t<275+1*365+60 |
         t>275+2*365 & t<275+2*365+60 | t>275+3*365 & t<275+3*365+60 |
         t>275+4*365 & t<275+4*365+60 | t>275+5*365 & t<275+5*365+60 |
         t>275+6*365 & t<275+6*365+60 | t>275+7*365 & t<275+7*365+60 |
         t>275+8*365 & t<275+8*365+60 | t>275+9*365 & t<275+9*365+60 |
         t>275+10*365 & t<275+10*365+60 | t>275+11*365 & t<275+11*365+60,
         500000, 0)}
bIfunc = function(bSA){
  bI = 0.5*bSA}
```

```
### EQUATIONS ===== ===== ===== ===== =====

## States (x) =====
SAIDW = function(t, x, pars){

  # dependent parameters
  temp = tempfunc(t)
  NC = NCfunc(temp)
  sigma = sigmafunc(NC)
  alpha = alphafunc(NC)
  gamma = gammafunc(NC)
  mSA = mSAfunc(t)
  mID = mIDfunc(t)
  bSA = bSAfunc(t)
  bI = bIfunc(bSA)

  with(as.list(c(x, pars)), {
    h = h_interp(t)
    dS = -mu*S - betaA*A*S - betaID*(I + D)*S -  betaW*W*S + mSA +
         p*gb*gJ*(bSA*(S + A) + bI*I) - h*S
    dA = -mu*A + betaA*A*S + betaID*(I + D)*S + betaW*W*S - sigma*A +
         gamma*I + gamma*D + mSA - h*A
    dI = -mu*I + sigma*A - gamma*I - alpha*I + mID
    dD = -mu*D + alpha*I - gamma*D - xi*D + mID
    dW = rhoA*A + rhoID*(I + D) + omega - eta*W
    return(list(c(dS, dA, dI, dD, dW)))
  })
}

## Adjoints (lambda) =====
Lambda = function(t, lambda, pars){
  state = sapply(1:length(pars$x_interp), function(i){pars$x_interp[[i]]
(t)})
  names(state) = c("S", "A", "I", "D", "W")

  # dependent parameters
  temp = tempfunc(t)
  NC = NCfunc(temp)
  sigma = sigmafunc(NC)
  alpha = alphafunc(NC)
  gamma = gammafunc(NC)
  mSA = mSAfunc(t)
  mID = mIDfunc(t)
  bSA = bSAfunc(t)
  bI = bIfunc(bSA)

  pars = c(pars, temp, NC, sigma, alpha, gamma, mSA, mID, bSA, bI)

  with(as.list(c(lambda, pars, state)), {
    h = h_interp(t)
    dlambdaS = -(P*h - C2*(betaA*A + betaID*(I + D) + betaW*W) +
                lambdaS*(-mu - betaA*A - betaID*(I + D) - betaW*W +
                p*gb*gJ*bSA - h) + lambdaA*(betaA*A + betaID*(I + D) +
```

```
                betaW*W))
     dlambdaA = -(P*h - C2*betaA*S + lambdaS*(-betaA*S + p*gb*gJ*bSA) +
                lambdaA*(-mu + betaA*S - sigma - h) + lambdaI*(sigma) +
                lambdaW*(rhoA))
     dlambdaI = -(-C2*betaID*S + lambdaS*(-betaID*S + p*gb*gJ*bI) +
                lambdaA*(betaID*S + gamma) + lambdaI*(-mu - gamma - alpha) +
                lambdaD*(alpha) + lambdaW*(rhoID))
     dlambdaD = -(-C2*betaID*S + lambdaS*(-betaID*S) + lambdaA*(betaID*S +
                gamma) + lambdaD*(-mu - gamma - xi) + lambdaW*(rhoID))
     dlambdaW = -(-C2*betaW*S + lambdaS*(-betaW*S) + lambdaA*(betaW*S) +
                lambdaW*(-eta))
     return(list(c(dlambdaS, dlambdaA, dlambdaI, dlambdaD, dlambdaW)))
   })
 }


 ### FORWARD-BACKWARDS SWEEP ===== ===== ===== ===== =====

 ## Initialization =====
 delta = 0.01
 test = -1
 count = 0

 # initial guesses
 h = rep(0, tlen)
 h_interp = approxfun(tvals, h, rule=2)
 pars = c(pars, h_interp)
 solx = ode(y=inits_sta, times=tvals, func=SAIDW, parms=pars)
 x_interp = lapply(2:ncol(solx), function(x){approxfun(solx[,c(1,x)],
             rule = 2)})
 pars = c(pars, x_interp)
 lambda = matrix(0, tlen, 5)

 ## While Loop =====
 while(test < 0 & count < 100){

   print(count)

   # set previous control, state, adjoint
   h_old = h
   x_old = solx
   lambda_old = lambda

   # interpolate control (h)
   pars$h_interp = approxfun(tvals, h, rule=2)

   # solve and interpolate states (x)
   solx = ode(y=inits_sta, times=tvals, func=SAIDW, parms=pars)
   pars$x_interp = lapply(2:ncol(solx), function(y){approxfun(solx[,c(1,y)],
                   rule=2)})
```

```
  S = solx[,"S"]
  A = solx[,"A"]
  I = solx[,"I"]
  D = solx[,"D"]
  W = solx[,"W"]

  # solve and interpolate adjoints (lambda)
  sollambda = ode(y=inits_lam, times=rev(tvals), func=Lambda, parms=pars)
  lambda = sollambda[nrow(lambda):1,]

  lambdaS = lambda[,"lambdaS"]
  lambdaA = lambda[,"lambdaA"]
  lambdaI = lambda[,"lambdaI"]
  lambdaD = lambda[,"lambdaD"]
  lambdaW = lambda[,"lambdaW"]

  # calculate control characteristic and update control
  h_char = with(pars, (P*(S + A) - lambdaS*S - lambdaA*A)/(2*C1))
  h_star = pmin(hmax, pmax(0, h_char))
  h = (h_old + h_star)*0.5

  # test convergence
  test = delta*sum(abs(h)) - sum(abs(h_old - h))
  print(test)

  # update count
  count = count + 1
}


### TOTAL COST (J) ===== ===== ===== ===== =====

ProfitH = sum(P*h*(S + A))*dt
CostH = sum(C1*h^{2})*dt
CostI = sum(C2*(betaA*A*S + betaID*(I + D)*S + betaW*W*S))*dt
tot = ProfitH - CostH - CostI

print(ProfitH)
print(CostH)
print(CostI)
print(tot)


### PLOTS ===== ===== ===== ===== =====

OC_states = solx %>% data.frame(S=S, A=A, I=I, D=D, W=W)
OC_har = data.frame(tvals, h)

SAID_plot = OC_states %>% gather(key, individuals, S, A, I, D) %>%
  ggplot(aes(x=time, y=individuals, color=key)) + geom_line() +
  ggtitle("(a) Market Pool") + xlab("days") + ylab("individuals") +
  theme(legend.position=c(1,1),legend.justification=c("right","top"),
        legend.margin=margin(5,5,5,5))
```

```
W_plot = OC_states %>% gather(key, individuals, W) %>%
  ggplot(aes(x=time, y=individuals, color=key)) + geom_line() +
  ggtitle("(b) Water Contamination") + xlab("days") + ylab("individuals") +
  theme(legend.position=c(1,1),legend.justification=c("right","top"),
        legend.margin=margin(5,5,5,5))

h_plot = OC_har %>% gather(key, har, h) %>%
  ggplot(aes(x=tvals, y=h, color=key)) + geom_line() +
  ggtitle("(b) Optimal Harvesting Rate") + xlab("days") +
  ylab("ind. per day") +
  theme(legend.position=c(1,1),legend.justification=c("right","top"),
        legend.margin=margin(5,5,5,5))

#plot_grid(SAID_plot, W_plot, h_plot, ncol = 1, nrow = 3)
plot_grid(SAID_plot, h_plot, ncol = 1, nrow = 2)
```