Optimal control of neglected tropical diseases using a simple dynamic model

Chris Hoover

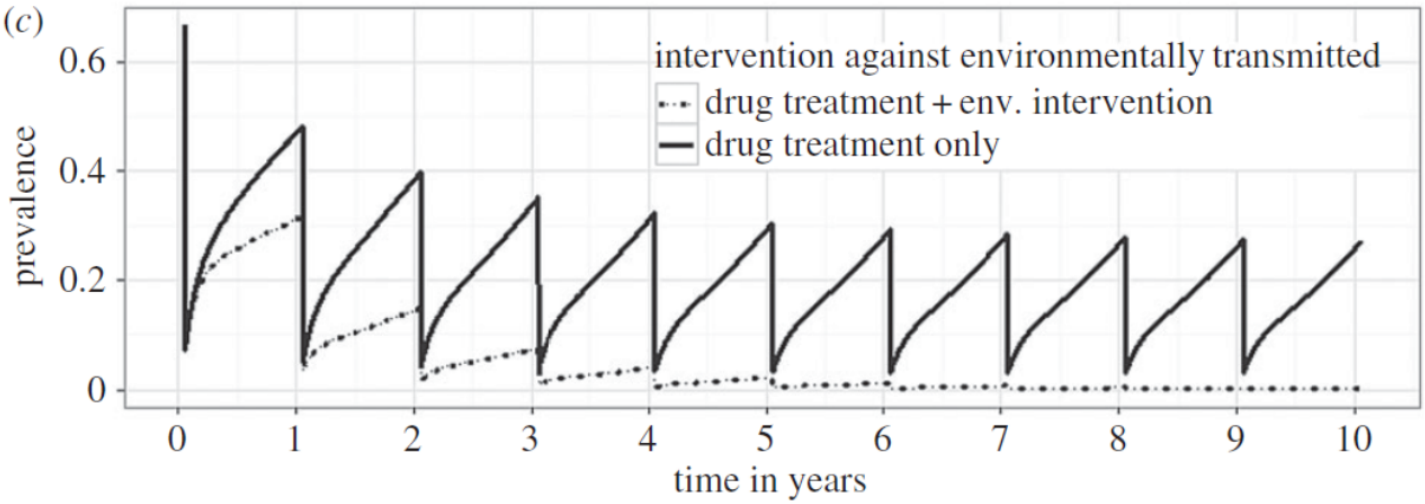
May 18, 2018

Neglected tropical diseases (NTDs) tend to be diseases of poverty that thrive in tropical, disadvantaged communities that often lack access to basic amenities such as clean water, sanitation, and healthcare. They are generally environmentally mediated meaning that some component of their transmission cycle depends on the environment (e.g. a vector, intermediate host, or free-living environmental stage). Control of NTDs generally relies on the administration of drugs that cure individuals from the disease and may confer short-term immunity. However, following drug administration, people are often reinfected by the environmental stage(s) of the pathogen that persist through mass drug administration (MDA) campaigns. Additional interventions that target these environmental reservoirs are often available, but underutilized due to their indirect effects on improving individuals' health.

Using a simple, generalizable model of NTD transmission presented in Garchitorena et al, two interventions will be considered: 1) drug administration, implemented as a pulse reduction in the state variable, I, that reduces the prevalence of the disease in the population and 2) environmental remediation (e.g. improvement in sanitation, vector control), implemented as a permanent alteration of a model parameter, that reduces the transmission of the disease.

Part 1: Reproduce Fig3c

The first goal is to code the model and reproduce Figure 3c (below) from Garchitorena et al comparing the effects of interventions that only target infected people, such as mass drug administration (MDA), to the effects of combined interventions that target infected people and the environment, such as sanitation, mosquito spraying, mollusciciding, etc.



The model consists of two state variables, I and W, that correspond to the prevalence of infection in the human population (i.e. proportion infected at time=t) and the degree of contamination of the environment with the disease-causing agent, respectively. We make the simplifying assumption that individuals can only be susceptible, S, or infected, I, meaning

Table 1: Parameter values and descriptions used in the model

	Value	Description
β_E	9.26e-05	Transmission rate from environment to human population
β_D	0.00555	Human to human transmission rate
γ	0.003	Rate of recovery from infected back to susceptible
Ω	0	Recruitment rate of infectious agents in the environment
V	1	Abundance of vectors/intermediate hosts/suitable environment
λ	1	Recruitment rate of infectious agents by infectious individuals
σ	1	Fraction of infectious agents produced that reach the environment
ρ	0.017	Mortality rate of infectious agents in the environment

$S + I = 1$ and eliminating the need for a recovered R compartment as is typical of SIR models but would complicate things here. The model equations then are:

$$\frac{dI}{dt} = (\beta_E W + \beta_D I)(1 - I) - \gamma I$$

$$\frac{dW}{dt} = \Omega + V\sigma\lambda I - \rho W$$

with parameter definitions and values in table 1 below

We start by making the simplifying assumption that there is no exogenous production of infectious agents ($\Omega = 0$) and that environment-human transmission is determined only by the transmission rate, β_E, with no role of vectors or intermediate hosts (i.e. $V = \sigma = \lambda = 1$). This is approximately equivalent to a simple Cholera model with the form:

$$\frac{dI}{dt} = (\beta_E W + \beta_D I)(1 - I) - \gamma I$$

$$\frac{dW}{dt} = I - \rho W$$

With this simplified model, we can get the equilibrium estimates of W and I:

$$\frac{dW}{dt} = I - \rho W$$

At equilibrium:

$$W^* = \frac{I}{\rho}$$

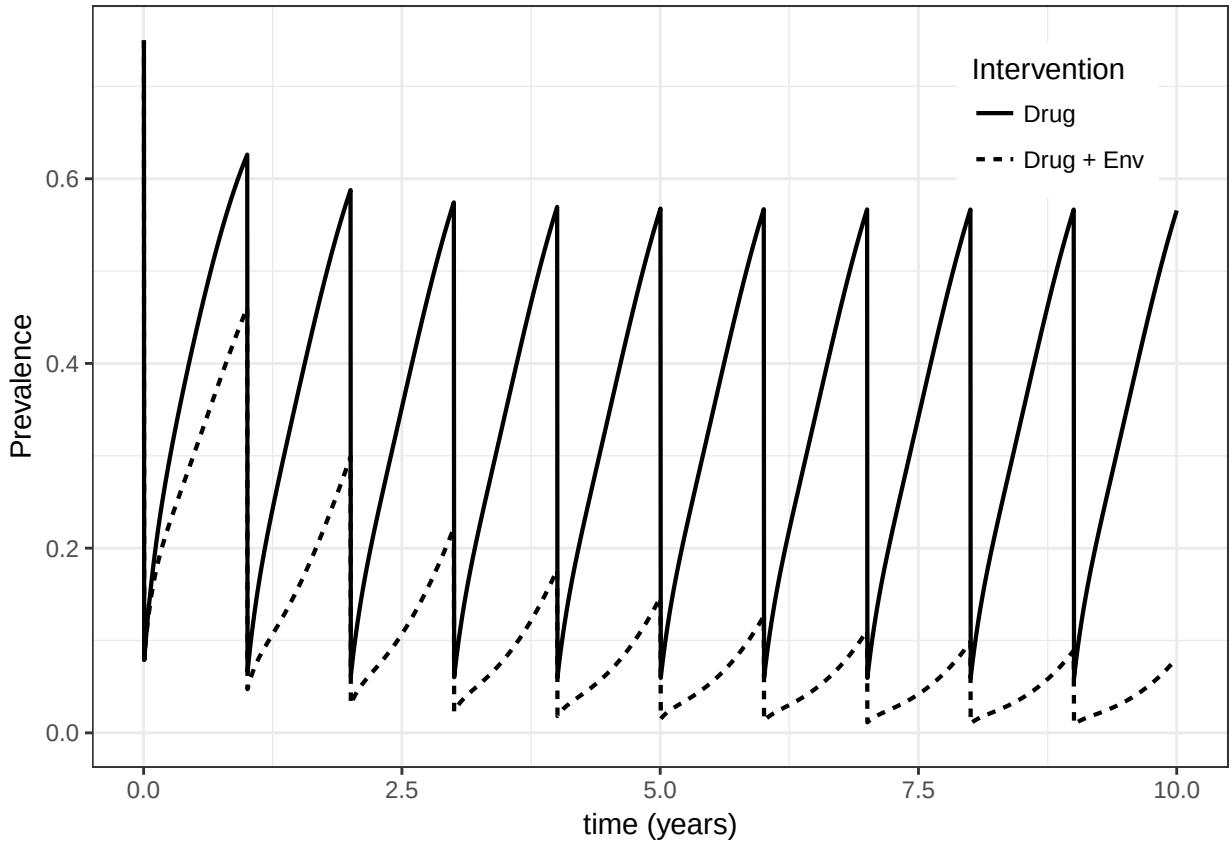
Substituting we get the following quadratic with solutions at $I^* = 0$ and $I^* =$ endemic equilibrium:

$$0 = \left(\frac{\beta_E I^*}{\rho} + \beta_D I^*\right)\left(1 - I^*\right) - \gamma I^*$$

Now we use these equilibrium estimates as starting values for the continuous time model and simulate ten years of interventions with the `deSolve` package to reproduce Fig 3c

```
#Parameter values from table 1 (some excluded) drawn from Fig S1 of the manuscript
pars <- c(beta_e = 9.26e-5, beta_d = 5.55e-3,
          gamma = 1/360, rho = 1/60)

#Function to get equilibrium prevalence (I) and equilibrium environmental contamination (W)
get_eq <- function(parameters){
  with(as.list(parameters),{
    init <- rootSolve::uniroot.all(function(I) ((beta_e*I / rho) + beta_d*I)*(1-I) - gamma*I,
                                   interval = c(0,1))
    eq_I <- init[which(init > 0)]
```

```
    eq_W <- eq_I/rho

    return(c(I = eq_I, W = eq_W))
  })
}

#Function to run model `t` days, with starting conditions `n` and parameter set `p`
simp_mod = function(t, n, p){
  with(as.list(p),{
    I = n[1]
    W = n[2]

    dIdt = (beta_e*W + beta_d*I)*(1-I) - gamma*I
    dWdt = I - rho*W

    return(list(c(dIdt, dWdt)))
  })
}

#Events representing interventions to implement in the model
n_years <- 10       #Simulate 10 years of annual intervention
drug_efficacy = 0.9 #90% reduction in prevalence at each drug treatment
run_time <- c(1:(365*n_years)) #Run model for 10 years

#data frame with event times for drug administration
drugs <- data.frame(var=rep('I', times = n_years),
                        time = c(0:(n_years-1))*365 + 1,
                        value = rep((1-drug_efficacy), times = n_years),
                        method = rep("mult", times = n_years))

#Run model with annual drug administration intervention
drug_only <- as.data.frame(ode(get_eq(pars), run_time, simp_mod, pars,
                               events = list(data = drugs)))

#Double mortality rate of environmental infectious agents for environmental intervention
pars_env <- pars
pars_env["rho"] <- pars["rho"] * 2

#Run model with annual drug administration and environmental intervention
drug_env <- as.data.frame(ode(get_eq(pars), run_time, simp_mod, pars_env,
                               events = list(data = drugs)))

#Plot results
rbind(drug_env, drug_only) %>%
  mutate(Intervention = c(rep("Drug + Env", length(run_time)),
                          rep("Drug", length(run_time)))) %>%
  ggplot(aes(x = time/365, y = I, lty = Intervention)) +
    theme_bw() + theme(legend.position = c(0.85,0.85)) +
    geom_line(size = 0.75) + xlab("time (years)") + ylab("Prevalence") + ylim(c(0,0.75))
```



Looks a little bit different from Fig 3c in that there's a higher starting prevalence and more rebound between drug treatments in the `Drug + Env` group. This is likely due to transmission parameters that are too high, so let's check that out.

```
#Function to get R0 estimate given model parameters
get_r0 <- function(parameters){
  with(as.list(parameters),{
    r0_d <- beta_d / gamma
    r0_e <- beta_e / (gamma*rho)

    r0_d + r0_e
  })
}

get_r0(pars)
```

```
## [1] 3.99816
```

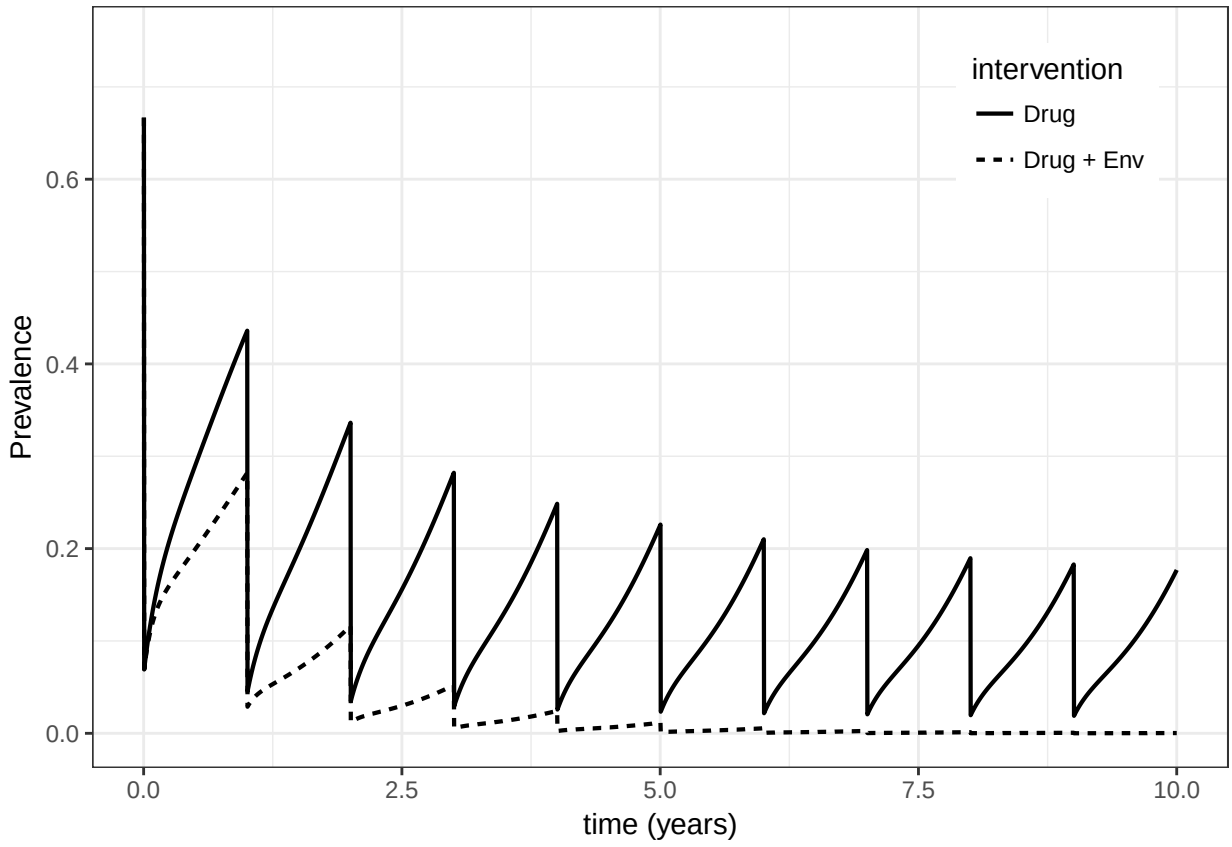
So $R_0 = 4$ with the parameters pulled from the SI, but Fig3c says $R_0 = 3$. Let's see if we can bring the transmission parameters down a bit and get closer to Fig3c.

```
r0 <- 3
pars2 <- pars

#Use equations from Fig S1 to estimate transmission parameters given desired r0 of 3
pars2["beta_d"] <- pars["gamma"]*r0 / 2
pars2["beta_e"] <- pars["gamma"]*r0*pars["rho"] / 2

get_r0(pars2)
```

```
## [1] 3
```

Now that we have the same R_0, let's try the figure again

```
# Rerun model with each intervention scenario and new parameter sets
drug_only2 <- as.data.frame(ode(get_eq(pars2), run_time, simp_mod, pars2,
                                events = list(data = drugs)))

pars_env2 <- pars2
pars_env2["rho"] <- pars2["rho"] * 2

drug_env2 <- as.data.frame(ode(get_eq(pars2), run_time, simp_mod, pars_env2,
                               events = list(data = drugs)))

#Plot results
rbind(drug_env2, drug_only2) %>%
  mutate(intervention = c(rep("Drug + Env", length(run_time)),
                          rep("Drug", length(run_time)))) %>%
  ggplot(aes(x = time/365, y = I, lty = intervention)) +
    theme_bw() + theme(legend.position = c(0.85,0.85)) +
    geom_line(size = 0.75) + xlab("time (years)") + ylab("Prevalence") + ylim(c(0,0.75))
```



Looks about right!

Part 2: Translate the model into a Markov Decision Process (MDP) solvable with stochastic dynamic programming (SDP)

Next we want to translate our system of continuous time differential equations into a Markov Decision Process (MDP) consisting of **1)** a Markov chain in which the state of the system at time $t+1$ is dependent only on the current state of the system (i.e. the state at t) and **2)** a decision or control action that is being made at each state transition (i.e. from t to $t+1$). We therefore want a single, discrete time equation that captures about the same dynamics of our simple disease system.

We'll again start with the assumption that the dynamics of the infectious agents in the environment are faster than the

dynamics of the prevalence in the human population, therefore they reach a steady state equilibrium:

$$W^* = \frac{I}{\rho}$$

Again, substituting we get:

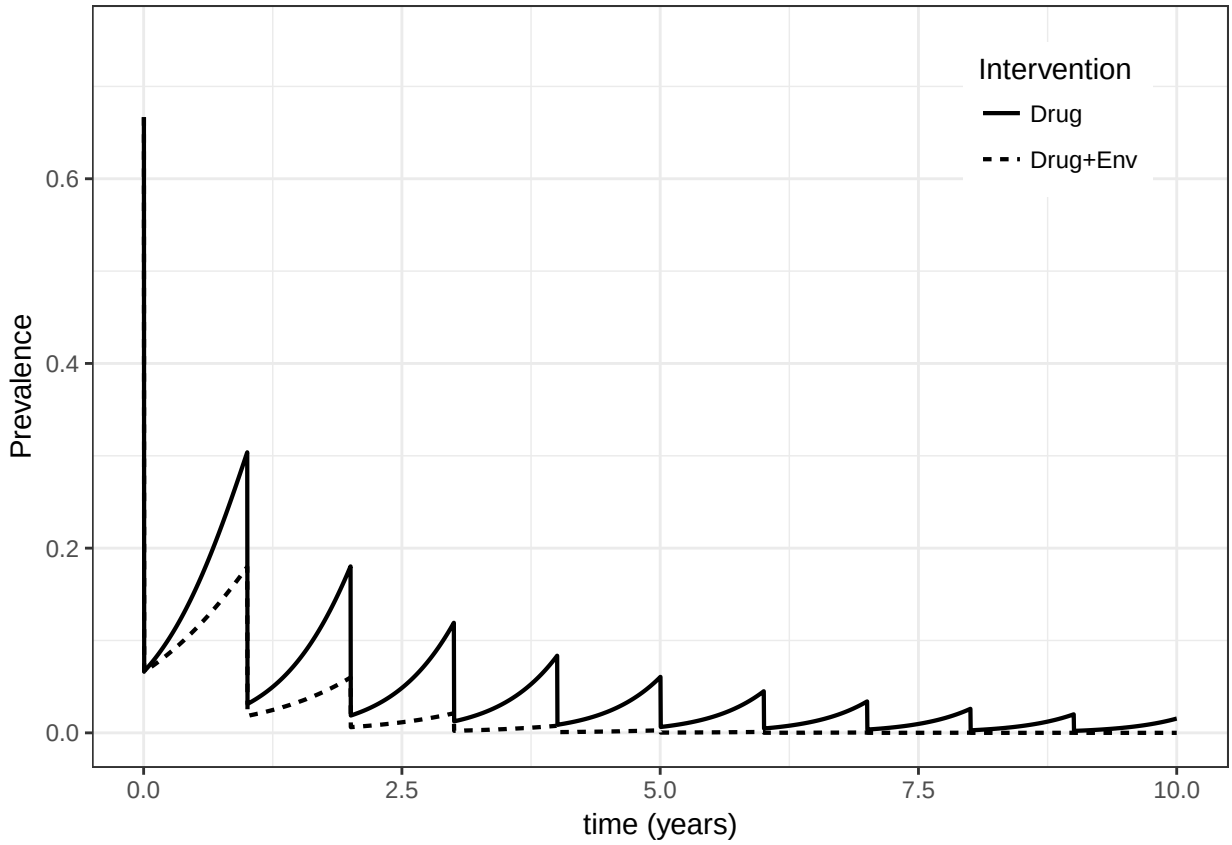
$$\frac{dI}{dt} = \Big(\frac{\beta_E I}{\rho} + \beta_D I\Big)\Big(1 - I\Big) - \gamma I$$

which can be translated to a discrete time model as:

$$I_{t+1} = I_t + \Big(\frac{\beta_E I_t}{\rho} + \beta_D I_t\Big)\Big(1 - I_t\Big) - \gamma I_t$$

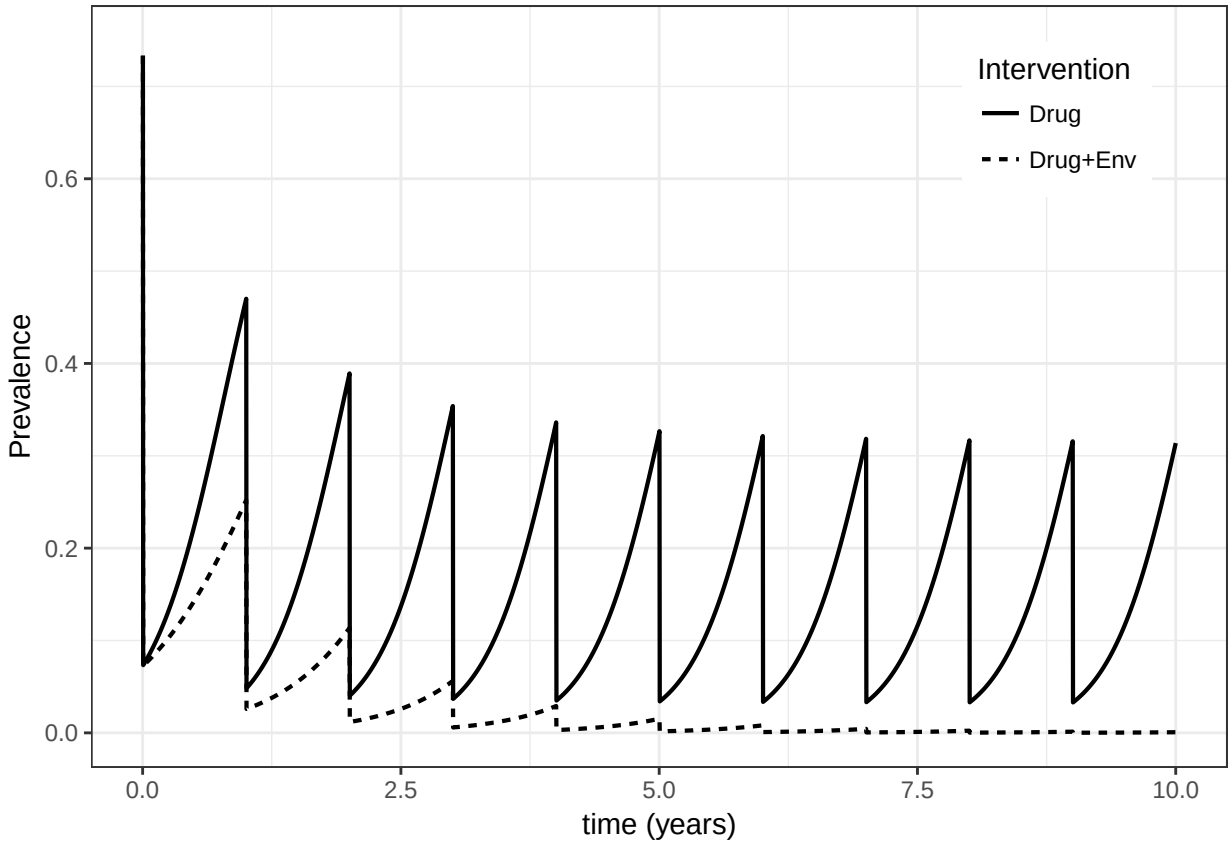
let's check to make sure that the dynamics of the continuous time model hold in this simplified, discrete time version of the model.

```
#Discrete time version of the mode with functionality for MDA action if MDA=1 and no effect if MDA = 0
discrete_mod <- function(I, parameters, MDA=0){
  with(as.list(parameters),{
    ((beta_e * I)/rho + beta_d * I)*(1-I) - gamma*I - MDA*drug_efficacy*I
  })
}

discrete_sim <- function(time, I_0_drug, I_0_env, model,
                         drug_parameters, env_parameters){

  #Initialize vectors to fill
  drug_fill <- vector("numeric", length = time)
    drug_fill[1] <- I_0_drug
  env_fill <- vector("numeric", length = time)
    env_fill[1] <- I_0_env

  #Run models over time with MDA occurring in same years as above in continuous time model
  for(t in 1:(time-1)){
    mda = ifelse(t %in% (c(0:(n_years-1))*365 + 1), 1, 0)

    drug_fill[t+1] <- drug_fill[t] + model(I = drug_fill[t], parameters = drug_parameters, MDA = mda)
    env_fill[t+1] <- env_fill[t] + model(I = env_fill[t], parameters = env_parameters, MDA = mda)
  }

  #Return results
  return(tibble(Time = rep(c(1:time),2),
                Intervention = c(rep("Drug", time), rep("Drug+Env", time)),
                Prevalence = c(drug_fill, env_fill)))
  }

test_discrete <- discrete_sim(time = length(run_time),
                              I_0_drug = get_eq(pars2)[1],
                              I_0_env = get_eq(pars2)[1],
                              model = discrete_mod,
                              drug_parameters = pars2,
                              env_parameters = pars_env2)

ggplot(test_discrete, aes(x = Time/365, y = Prevalence, lty = Intervention)) +
       theme_bw() + theme(legend.position = c(0.85,0.85)) +
       geom_line(size = 0.75) + xlab("time (years)") + ylab("Prevalence") + ylim(c(0,0.75))
```



It looks like our MDA intervention packs a bit more punch in this version of the model. That makes sense since we've simplified the environmental dynamics. To compensate, lets bump up the transmission parameter a bit.

```
#Bump up environment to human transmission parameter by 50%
  pars3 <- pars2
    pars3["beta_e"] <- pars2["beta_e"] * 1.5
  pars_env3 <-pars_env2
    pars_env3["beta_e"] <- pars_env2["beta_e"] * 1.5

test_discrete2 <- discrete_sim(time = length(run_time),
                               I_0_drug = get_eq(pars3)[1],
                               I_0_env = get_eq(pars3)[1],
                               model = discrete_mod,
                               drug_parameters = pars3,
                               env_parameters = pars_env3)

ggplot(test_discrete2, aes(x = Time/365, y = Prevalence, lty = Intervention)) +
       theme_bw() + theme(legend.position = c(0.85,0.85)) +
       geom_line(size = 0.75) + xlab("time (years)") + ylab("Prevalence") + ylim(c(0,0.75))
```



That looks closer to what we started with, let's continue.

Now we'll work through the "Six steps of stochastic dynamic programming" as described in Marescot et al to get our MDP and set ourselves up for analysis

1. **Define the optimization objective of the problem.**
   Our objective is to minimize the costs, C, associated with 1) infection of individuals and 2) implementing interventions to reduce their infections

2. **Define the set of states that represent the configuration of the system at time t**
   We define the state variable, X_t, as the prevalence of infection in the human population, $I_t \in [0, 1]$.

3. **Define the decision variable, A_t that is the component of the system to be controlled to meet the objective**
   We define the decision variable A_t as the proportion of capital, M, committed to drug administration with the rest allocated towards environmental interventions.

4. **Build a transition model describing the system's dynamics as a function of the decision variable and the system state in the prior time step**
   Beginning with the discrete time model above, we have:

   $$I_{t+1} = I_t + \Big(\frac{\beta_E I_t}{\rho} + \beta_D I_t\Big)\Big(1 - I_t\Big) - \gamma I_t$$

   Incorporating interventions based on our decision variable, A_t gives us:

   $$I_{t+1} = I_t + \Big(\frac{\beta_E I_t}{\rho + M(1 - A_t)\mu} + \beta_D I_t\Big)\Big(1 - I_t\Big) - I_t(\gamma + M A_t \theta)$$

   Where θ is a constant that converts capital, M to units of prevalence and μ is a constant that converts M to the same units as the mortality rate of infectious agents in the environment.

5. **Define the utility function $U_t(X_t, A_t)$ representing the desirability of acting in a given state**

   $$\max_A C = \max_A \sum_0^T \frac{-\Pi(I_t)}{\delta_t}$$

where

$$\Pi(I_t) = dI_t$$

d is the cost associated with having prevalence of I_t and A_t is the proportion of that capital allocated towards drug administration (with the rest allocated towards environmental intervention). We're maximizing the negative costs, aka minimizing the costs.

6. **Determine the optimal solution of the optimization problem**
   Step 6 is Part 3, see below

Part 3: Find the optimal solution using SDP in the `MDPtoolbox` R package

Let's translate the problem outlined above into code, borrowing from the wolf culling example in Marescot et al

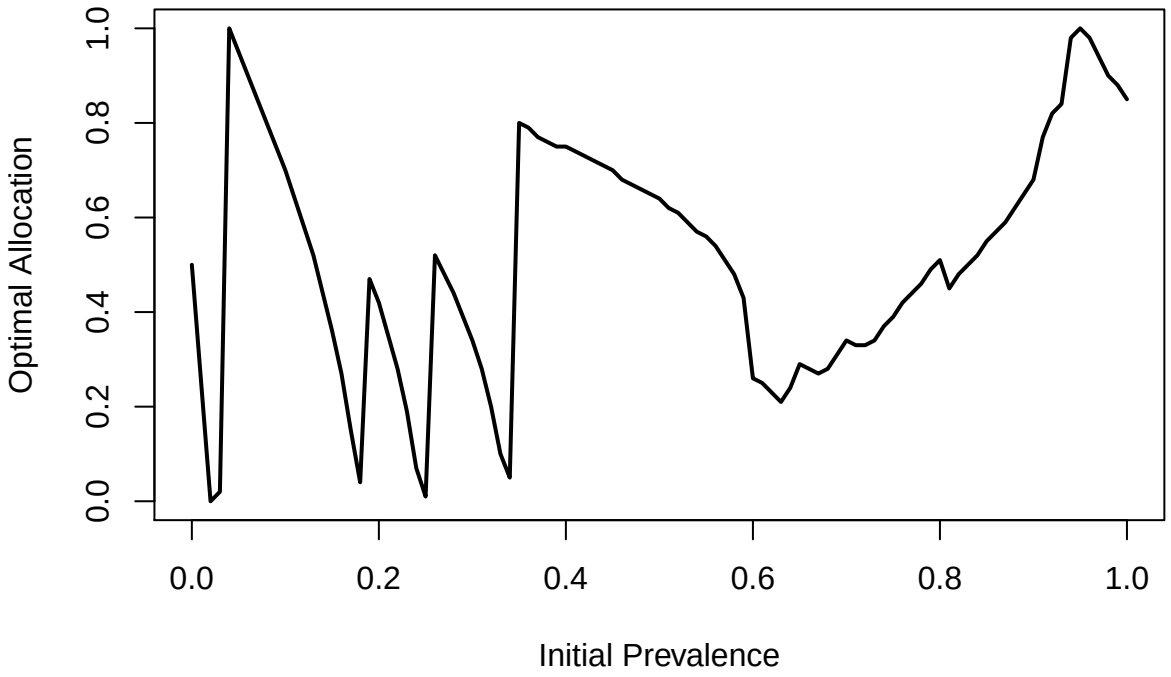
```
#Vector of all possible states, i.e. prevalence ranging from 0 to 1
states <- seq(0, 1, 0.01)

#Vector of possible actions, A_t, ranging from total investment in drugs (M=1)
#  to total investment in environmental intervention (M=0)
A <- seq(0, 1, 0.01)

#Parameters for the model and utility function:
mdp_pars <- c(pars3,
              "d" = 100,       #Arbitrary cost of having prevalence of I_t
              "M" = 1,         #Arbitrary capital available to spend on MDA
              #Somewhat arbitrary scaling of capital spent on MDA to reduction in prevalence,
              #  if M=1, 90% reduction (just as in simulations above)
              "theta" = 0.9,
              #If all capital spent on environmental intervention, double the environmental mortality rate
              "mu" = as.numeric(pars_env2["rho"]),
              "delta" = 0.9)  #Arbitrary discounting rate

#Dynamic model for MDP. We're modeling yearly interventions,
# so actually want to take action A on I and then consider I+365
mdp_1yr <- function(A_t, I_0, time = 365, model = mdp_mod, pars = mdp_pars, M=1){
  with(as.list(pars),{


  #Initialize vector to fill
  I_vec <- vector("numeric", length = time+1)
    I_vec[1] <- I_0

  #Run model over time with MDA occurring in first time step
  #  and environmental intervention permanent over the year
    a_env <- 1-A_t   #Capital invested in environment, acts through the whole year

  for(t in 1:(time)){
    a_mda <- ifelse(t == 1, A_t, 0)  #MDA occurs in first time step
    I_vec[t+1] <- I_vec[t] +
                  ((beta_e * I_vec[t])/(rho+a_env*M*mu) + beta_d * I_vec[t])*(1-I_vec[t]) -
                  I_vec[t]*(gamma + M*a_mda*I_vec[t]*theta)
  }

  #Return I_t at t=365 post implementation of A_t
  return(I_vec[time])
  })
}

#Define utility function
get_util <- function(x){
```

```
  with(as.list(mdp_pars),{
    -d*x
  })
}

#Transition matrix for I states and A actions
transition <- array(0, dim = c(length(states), length(states), length(A)))

#Utility matrix
utility <- array(0, dim = c(length(states), length(A)))

# Fill in the transition and utility matrices
# Loop on all states
# TODO: make this Tidy
for (i in 1:length(states)) {

    # Loop on all actions
    for (a in 1:length(A)) {

# Calculate the transition state at the next step, given the current state i and the intervention, a
        nextpop <- round(mdp_1yr(A_t = A[a], I_0 = states[i]), 2)
        nextind <- which(states == nextpop)

#Since this model is deterministic, assign probability of 1 to value of I_t+1, everything else =0
        transition[i, nextind, a] <- 1

# Compute utility
        utility[i,a] <- get_util(nextpop)

    } # end of action loop
} # end of state loop
```

Now we have the transition and utility matrices. The transition matrix indicates for a given starting prevalence, I_t, and action A_t, the probability of the state at I_{t+1}. Since this model is deterministic, the transition matrix contains a whole bunch of 0s and some 1s that indicate the value $I_{t+1}|I_t, A_t$. Future models will incorporate stochasticity in transmission to spread the probability around. The utility matrix simply represents the utility of each action, A_t applied to each starting prevalence, I_t.

```
# THis code is adopted from the wolf culling example in the Marescot et al paper
# it doesn't actually use MDPtoolbox directly, but manually does the same thing

# Discount factor
discount <- mdp_pars["delta"]

# Action value vector at tmax
Vtmax <- numeric(length(states))

# Action value vector at t and t+1
Vt <- numeric(length(states))
Vtplus <- numeric(length(states))

# Optimal policy vector
D <- numeric(length(states))

# Finite time horizon of 20 years
Tmax <- 20

# The backward iteration consists in storing action values in the vector Vt which is the maximum of
# utility plus the future action values for all possible next states. Knowing the final action
```

```
# values, we can then backwardly reset the next action value Vtplus to the new value Vt. We start
# The backward iteration at time T-1 since we already defined the action value at Tmax.
for (t in (Tmax-1):1) {

# We define a matrix Q that stores the updated action values for #all states (rows)
    # actions (columns)
    Q <- array(0, dim=c(length(states), length(A)))

    for (i in 1:length(A)) {

# For each capital allocation proportion we fill for all states values (row)
# the ith column (Action) of matrix Q
# The utility of the ith action recorded for all states is
# added to the product of the transition matrix of the ith
# action by the action value of all states
        Q[,i] <- utility[,i] + discount*(transition[,,i] %*% Vtplus)

    } # end of loop

    # Find the optimal action value at time t is the maximum of Q
    Vt <- apply(Q, 1, max)

# After filling vector Vt of the action values at all states,
# we update the vector Vt+1 to Vt and we go to the next step standing
# for previous time t-1, since we iterate backward
    Vtplus <- Vt

} # end of the time loop

# Find optimal action for each state
for (i in 1:length(states)) {
# We look for each state which column of Q corresponds to the
# maximum of the last updated value
# of Vt (the one at time t+1). If the index vector is longer than 1
# (if there is more than one optimal value we chose the median capital allocation rate)
    D[i] <- A[(median(which(Q[i,] == Vt[i])))]

}

plot(states, D, xlab="Initial Prevalence", ylab="Optimal Allocation", type = 'l', lwd = 2)
```



This isn't exactly what we were expecting, let's do some debugging. MDPtoolbox has a function to check that our transition and utility matrices are formatted properly and contain the right kind of data, let's use that

```
mdp_check(transition, utility)
```

```
## [1] "MDP Toolbox ERROR: Row sums of the matrix must be 1"
```

So looks like some rows in our transition matrix are not summing to 1. Let's try and figure out why.

```
#Get sum of rows in the slice of the array in which A_t = 50
rowSums(transition[,,50])
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
# Get indices of states vector that returns rowsum of 0 rather than 1
bad_vec <- which(rowSums(transition[,,50]) == 0)

#Run the mdp function with states that return rowsum=0 and try and match it to indices in states vector
# this is what is being done when building the transition matrix
bad_nextpops <- map2_dbl(rep(0.5, length(bad_vec)), states[bad_vec], mdp_1yr)
round(bad_nextpops, 2)
```

```
## [1] 0.41 0.47 0.57 0.57 0.57 0.57 0.57 0.57
```

```
which(states == round(bad_nextpops, 2))
```

```
## integer(0)
```

```r
#So for some reason, rounding the output from the mdp_1yr function to two digits
#and getting the corresponding vector index in states is not working for these values
#Value of nextpop
round(bad_nextpops[1], 2)
```

```
## [1] 0.41
```

```r
#Confirmed it is 0.41
round(bad_nextpops[1], 2) == 0.41
```

```
## [1] TRUE
```

```r
#Value of states
states[42]
```

```
## [1] 0.41
```

```r
#Confirm is is 0.41
states[42] == 0.41
```

```
## [1] FALSE
```

```r
#WHAT???

#Try another value from states
states[41]
```

```
## [1] 0.4
```

```r
states[41] == 0.40
```

```
## [1] TRUE
```

```r
# THIS works???

# So that is VERY STRANGE... but found the function "pmatch" that seems to alleviate this
# by converting things to characters and then matching them
pmatch(round(bad_nextpops[1], 2), states)
```

```
## [1] 42
```

```r
#Transition matrix for I states and A actions
transition2 <- array(0, dim = c(length(states), length(states), length(A)))

#Utility matrix
utility2 <- array(0, dim = c(length(states), length(A)))

# Fill in the transition and utility matrices
# Loop on all states
# TODO: make this Tidy
for (i in 1:length(states)) {

    # Loop on all actions
    for (a in 1:length(A)) {
```

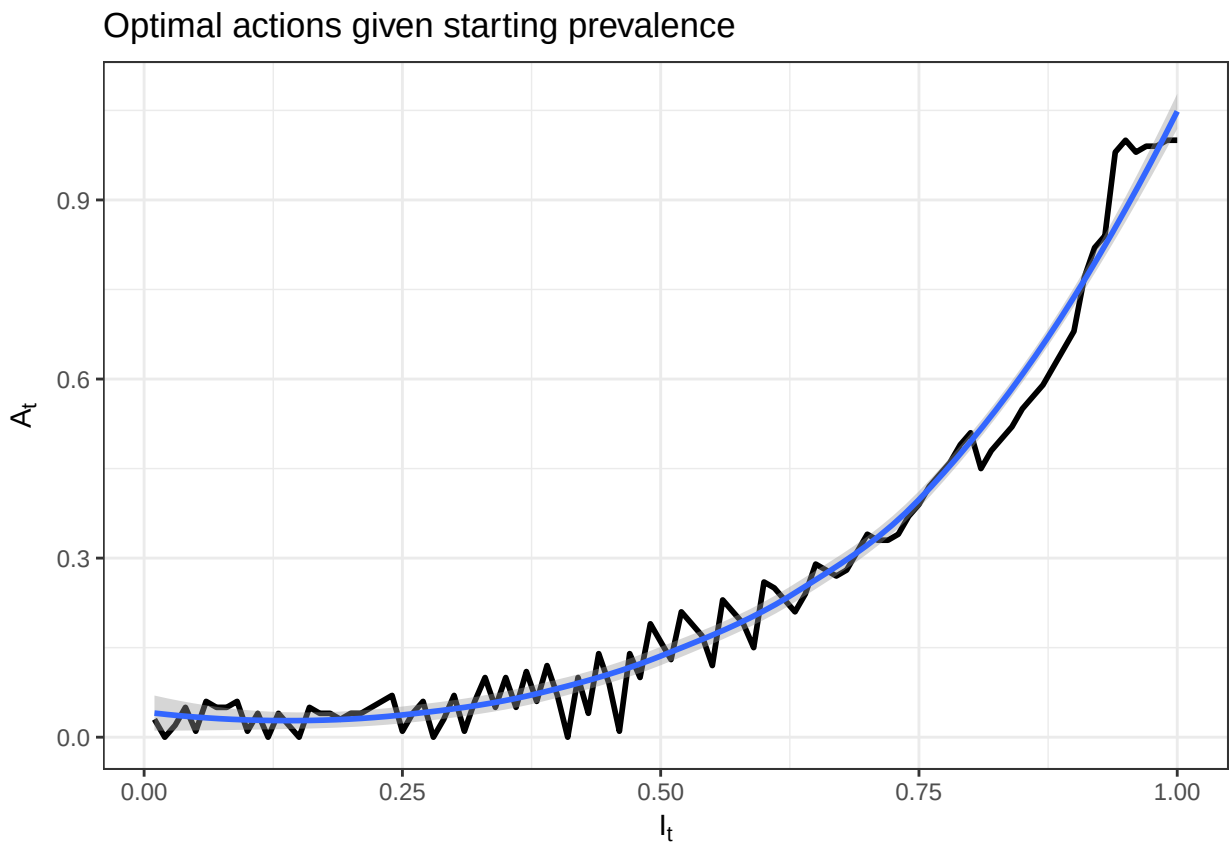
```
# Calculate the transition state at the next step, given the current state i and the intervention, a
        nextpop <- round(mdp_1yr(A_t = A[a], I_0 = states[i]), 2)
        nextind <- pmatch(nextpop, states) #replace which[] with pmatch

#Since this model is deterministic, assign probability of 1 to value of I_t+1, everything else =0
        transition2[i, nextind, a] <- 1

# Compute utility
        utility2[i,a] <- get_util(nextpop)

    } # end of action loop
} # end of state loop

# Run mdp check
mdp_check(transition2, utility2)
```

```
## [1] ""
```

```
#No error which means all our rows sum to 1. Woohoo!
```

```
# Action value vector at tmax
Vtmax2 <- numeric(length(states))

# Action value vector at t and t+1
Vt2 <- numeric(length(states))
Vtplus2 <- numeric(length(states))

# Optimal policy vector
D2 <- numeric(length(states))

for (t in (Tmax-1):1) {

# We define a matrix Q that stores the updated action values for #all states (rows)
    # actions (columns)
    Q2 <- array(0, dim=c(length(states), length(A)))

    for (i in 1:length(A)) {

# For each capital allocation proportion we fill for all states values (row) #the ith column (Action) of matri
# The utility of the ith action recorded for all states is
# added to the product of the transition matrix of the ith
# action by the action value of all states
        Q2[,i] <- utility2[,i] + discount*(transition2[,,i] %*% Vtplus2)

    } # end of the harvest loop

    # Find the optimal action value at time t is the maximum of Q
    Vt2 <- apply(Q2, 1, max)

# After filling vector Vt of the action values at all states,
# we update the vector Vt+1 to Vt and we go to the next step standing
# for previous time t-1, since we iterate backward
    Vtplus2 <- Vt2

} # end of the time loop

# Find optimal action for each state
for (i in 1:length(states)) {
# We look for each state which column of Q corresponds to the #maximum of the last updated value
```

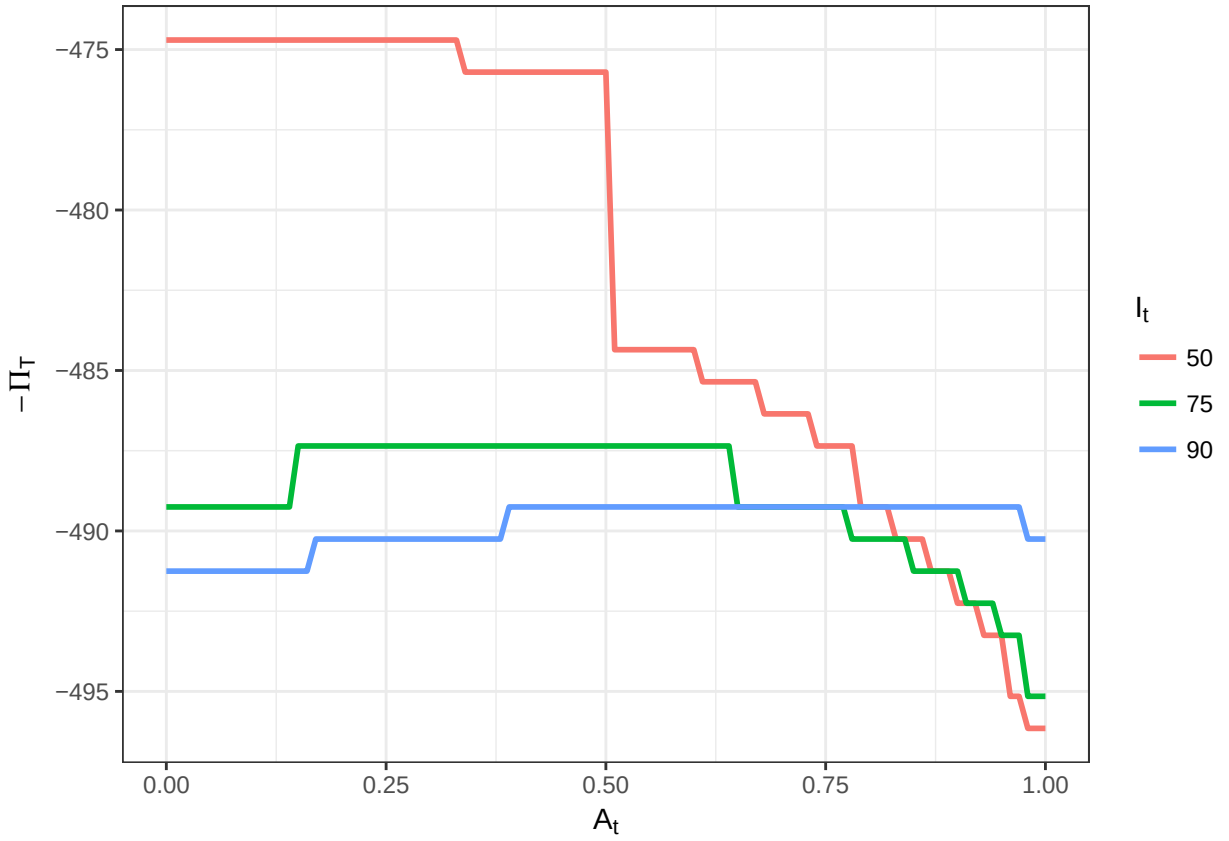
```
# of Vt (the one at time t+1). If the index vector is longer than 1 #(if there is more than one optimal value
    D2[i] <- A[(median(which(round(Q2[i,],1) == round(Vt2[i], 1))))]
    #print(pmatch(Vt2[i], Q2[i,]))
      #A[(min(which(Q2[i,] == Vt2[i])))]
      #
}
# Remove 0 starting prevalence since 0 will remain at 0 and any intervention will be "optimal"
as.data.frame(cbind(D2 = D2[-1], states = states[-1])) %>%
ggplot(aes(x = states, y = D2)) + geom_line(size = 1) +
    theme_bw() + ylab(expression(A[t])) + xlab(expression(I[t])) + stat_smooth() +
    ggtitle("Optimal actions given starting prevalence")
```



Cool! The general trend makes sense: at low starting prevalence, we minimize prevalence over twenty years by reducing transmission via investment in environmental remediation. When prevalence is high, though, this environmental remediation must be supplemented with MDA to effectively reduce prevalence. This is in line with the findings of Garchitorenea et al as well: if all we do is annual MDA, we end up with those "sawtooth" curves in which prevalence inevitably bounces back after treatment, but with some environmental intervention added in, we can reduce that rate of bounce back and make progress towards elimination.

We can also explain the jagged line by looking at some of the results a bit more closely:

```
#Each row of Q2 contains the output of our utility function summed over the time horizon (20 years)
# for a given starting prevalence and each of the 100 possible valeus of A_t.
data.frame(a_t = rep(A, 3),
           i_0 = rep(c(50, 75, 90), each = length(A)),
           Max_ut = c(Q2[51,], Q2[76,], Q2[91,])) %>%
  ggplot(aes(x = a_t, y = Max_ut, col = as.factor(i_0))) + geom_line(size = 1) +
    theme_bw() + guides(col=guide_legend(title=expression(I[t]))) + xlab(expression(A[t])) + ylab(expression(-
```



This shows that there's not much variability in which action, A_t is best given a starting prevalence I_t, i.e. for starting prevalence of $I - t = 90\%$, values of A_t between about 0.33 through about 0.97 all have the same total utility and the median value of A_t which achieves this utility is what's plotted in the Optimal Allocation plot. Changing our time horizon to be something like 100 years rather than 20 may smooth this out by adding more variability in how the interventions affect summed prevalence over time, but 20 years is a reasonable time horizon to shoot for disease elimination, so we'll stick with it.