Model fit

Chris Hoover

November 26, 2018

This document describes the epidemiological model, data, and fitting procedure used to derive estimates of transmission parameters for a baseline model used to estimate the effect of agrochemical pollution on schistosomiasis transmission

Initial exploration

First estimate R_0 over a range of potential values of β and λ to get a general idea of ranges to include in fitting procedure

```
test_betas <- seq(0.05,1.5,0.05)*init_pars["beta"]
test_lambdas <- seq(0.05,1.5,0.05)*init_pars["lambda"]

test_trans_pars <- expand.grid(beta = test_betas,
                               lambda = test_lambdas)

test_pars <- init_pars
test_pars["mu_P"] <- 0.9 # Kill predators for pred free estimation

test_r0_beta_lambda <- function(beta, lambda){
  test_pars["beta"] <- beta
  test_pars["lambda"] <- lambda

  r0.Ag(parameters = test_pars)[3]
}

test_trans_pars$r0 <- mapply(test_r0_beta_lambda,test_trans_pars$beta, test_trans_pars$lambda)

#Plot heatmap
test_trans_pars %>%
  ggplot(aes(x = beta, y = lambda, fill = r0, z = r0)) +
    geom_tile() +
    geom_contour(breaks = c(1), col = "white") +
    scale_fill_viridis(option = "magma", direction = 1) +
    labs(y = expression(lambda),
         x = expression(beta)) +
    theme_bw() +
    theme(axis.ticks=element_blank(),
          axis.text=element_text(size=12),
          axis.title=element_text(size=14),
          legend.title=element_text(size=15),
          legend.text=element_text(size=12)) +
    guides(fill=guide_legend(title=expression(R[0]),
                             reverse = T))  +
    annotate(geom = "point", x = init_pars["beta"], y = init_pars["lambda"], col = "white")
```

We'll stick witht the β estimate from previous work as we don't have sufficient snail data to fit to. This leaves us with the task of fitting worm burden model outputs to human infection data using the snail to man transmission parameter, λ. In previous work, we tried fitting with a single parameter and were unable to adequately fit the model due to highly seasonal transmission. We therefore divided transmission into low and high seasons and fit the model assuming a λ_{lo} and λ_{hi} were operative during low and high transmission seasons, respectively. We'll try both of those methods here as well as fitting a seasonal model that incorporates seasonality in the snail population.

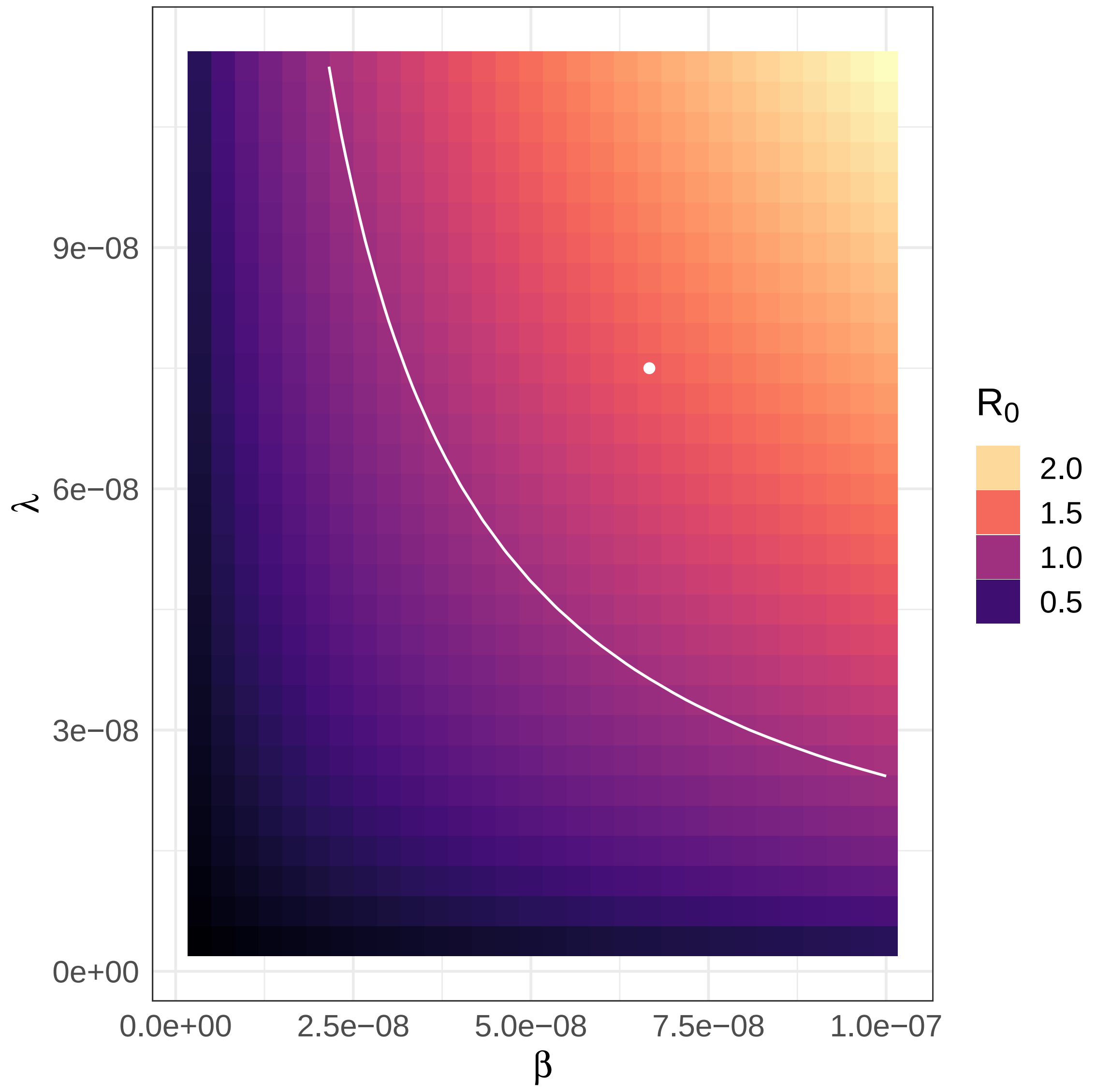


Figure 1: Line indicates R0=1 and dot indicates coordinates (and R0 value) of initial parameter set

Gather epi data to fit to

Data comes from community in Senegal where prawn intervention was first piloted. This data comes from the control community where prawns were not introduced. Data and fitting routine are the same as reported in Arakala, Hoover, et al., but this model contains additional parameters in order to accomodate a range of potential agrochemcial influences

```
N <- 129 #number of people in community who were treated
H <- 300

EggToWormConvert <- 3.6 #convert egg burden to worm burden based on estimate of eggs/mated female from Cheever s
cvrg <- N/H #43% coverage estimated in this community 129 out of 300 people

init_pars["H"] <- H
init_pars["cvrg"] <- cvrg

#Baseline (Feb 2012) estimates
  W_baseline<-6.5/EggToWormConvert
  W_baseline_k<-0.08
  W_baseline_sd<-sqrt( (W_baseline)+((W_baseline^2)/W_baseline_k ) )
  W_baseline_se<-W_baseline_sd/sqrt(N)

#5 month (July 2012) epi data point estimates
  W_5month_field_mean<-1.5/EggToWormConvert
  W_5month_field_k<-0.02
  W_5month_field_sd<-sqrt( (W_5month_field_mean)+((W_5month_field_mean^2)/W_5month_field_k ) )
  W_5month_field_se<-W_5month_field_sd/sqrt(N)

#Feb 13 epi data estimates (in the middle of high transmission season)
  W_Feb13_field_mean<-161/EggToWormConvert
  W_Feb13_field_k<-0.21
  W_Feb13_field_sd<-sqrt( (W_Feb13_field_mean)+((W_Feb13_field_mean^2)/W_Feb13_field_k ) )
  W_Feb13_field_se<-W_Feb13_field_sd/sqrt(N)

#Sept 13 epi data estimates (in the middle of high transmission season)
  W_Sep13_field_mean<-17.6/EggToWormConvert
  W_Sep13_field_k<-0.29
  W_Sep13_field_sd<-sqrt( (W_Sep13_field_mean)+((W_Sep13_field_mean^2)/W_Sep13_field_k ) )
  W_Sep13_field_se<-W_Sep13_field_sd/sqrt(N)

#Timepoints of epi datapoint collection
  timepoints<-c(1, 156, 375, 594)
  traj_time <- c(1:(max(timepoints) +2))    #time vector to run over

#Estimates of measured worm burden at epi timepoints
  measured_ws <- c(W_baseline, W_5month_field_mean, W_Feb13_field_mean, W_Sep13_field_mean)
  w_errors <- c(W_baseline_se, W_5month_field_se, W_Feb13_field_se, W_Sep13_field_se)
  measured_ks <- c(W_baseline_k, W_5month_field_k, W_Feb13_field_k, W_Sep13_field_k)

epi_plot <- as.data.frame(cbind("time" = timepoints,  "Worm_burden" = measured_ws, "se" = w_errors))

epi_plot %>%
  ggplot(aes(x = timepoints, y = measured_ws)) + geom_point() + theme_bw() + ylim(0,60) +
    geom_errorbar(ymin = measured_ws - w_errors, ymax = measured_ws + w_errors, width = 5)
```

```
#timepoints at which transmission shifts from low to high
  seasons <- c(1, 134, 263, 502)

#Events data frame containing timepoints at which MDA occurs
  mdas <- data.frame(var = rep('Wt.cvrg', 4),
                     time = c(2, 22, 157, 376),
```

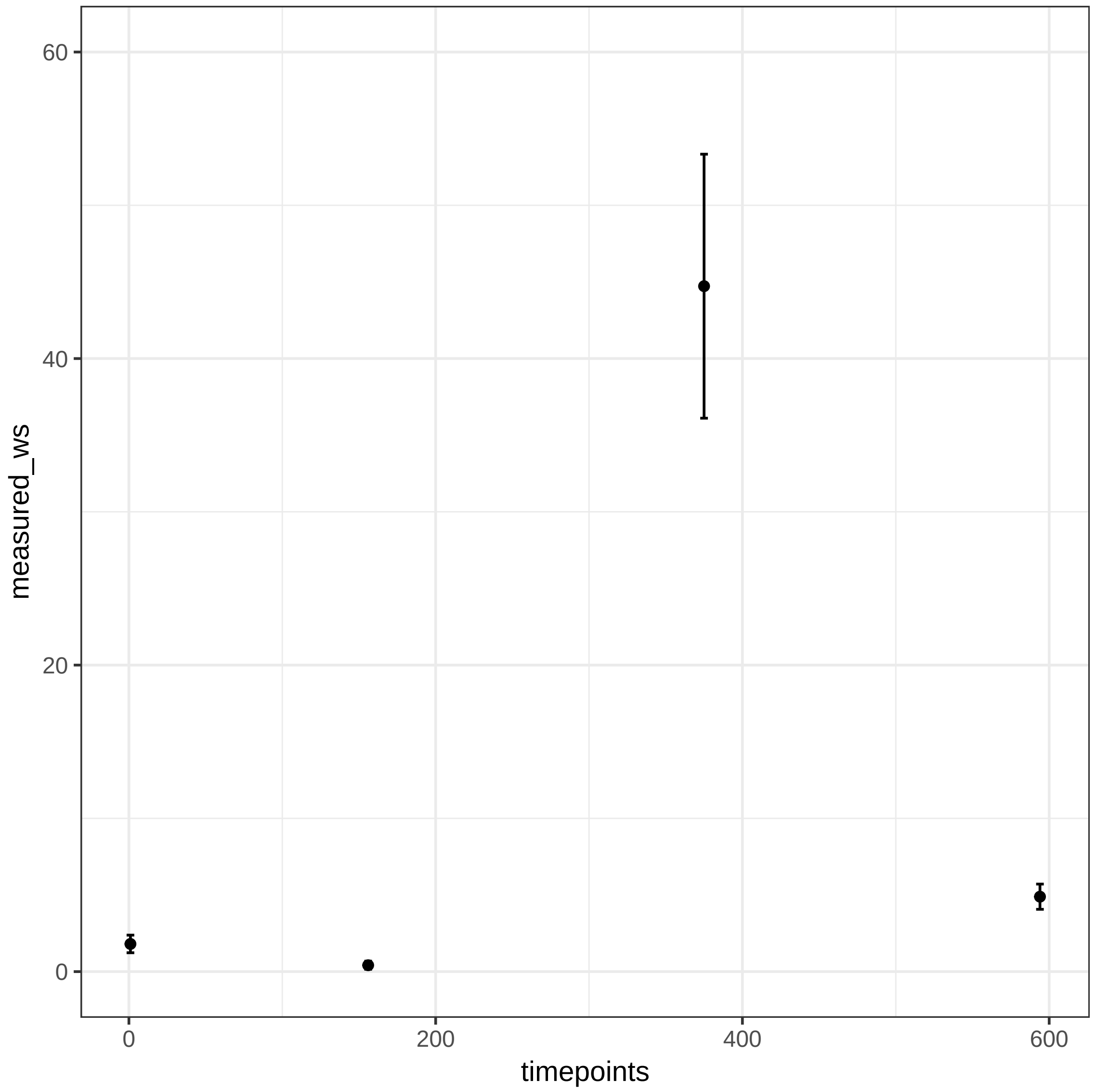


Figure 2: Estimates of mean worm burden in Senegal community to fit model to

```
                      value = rep(0.06, 4),      #94% efficacy
                      method = rep('mult', 4))
```

Fit model to epi data

Single transmission parameter

```
#Somewhat arbitrary starting conditions to initiate the model
cvrg <- init_pars["cvrg"]

base_mda.start = c(S = 40*area,
                   E = 0,
                   I = 0,
                   Wt = 2*cvrg,
                   Wu = 2*(1-cvrg),
                   P = 0)

params <- init_pars
params["mu_P"] <- 0.9 # Kill predators for pred free estimation

#Function to simulate transmission, given single transmission parameter and return negative log likelihood from
Tx_simulate_1par<-function(lambda){

  params["lambda"] <- lambda

    eqbm_vals <- runsteady(y = base_mda.start, func = agrochem_mda_mod, parms = params)$y

    traj <- data.frame(ode(eqbm_vals, traj_time, agrochem_mda_mod,
                           params, events = list(data = mdas)))

    modeled <- traj$Wt[timepoints + 2] * 0.5 * mapply(phi_Wk, traj$Wt[timepoints + 2], measured_ks)

    point_LL <- mapply(pointLogLike, measured_ws, w_errors, modeled)

    point_LL[!is.finite(point_LL)] <- NA

    logLike <- sum(point_LL, na.rm = TRUE)

    return(-logLike)

}

#Fit using Optimize for 1-dimensional optimization
  op_min_1par <- optimize(Tx_simulate_1par, interval = c(init_pars["lambda"]/100, init_pars["lambda"]*10), tol =

#Extract best fitting parameter
  bestLambda_1par <- op_min_1par$minimum

  fit_pars_1par <- init_pars
  fit_pars_1par["lambda"] <- bestLambda_1par
  fit_pars_1par["mu_P"] <- 0.9  # Want to estimate predator-free R0

#simulate with best fitting parameter
  fit_eq_vals_1par <- runsteady(y = base_mda.start, func = agrochem_mda_mod,
                                parms = fit_pars_1par)$y

  fit_traj_1par <- data.frame(ode(fit_eq_vals_1par, traj_time, agrochem_mda_mod,
                                  fit_pars_1par, events = list(data = mdas)))
```

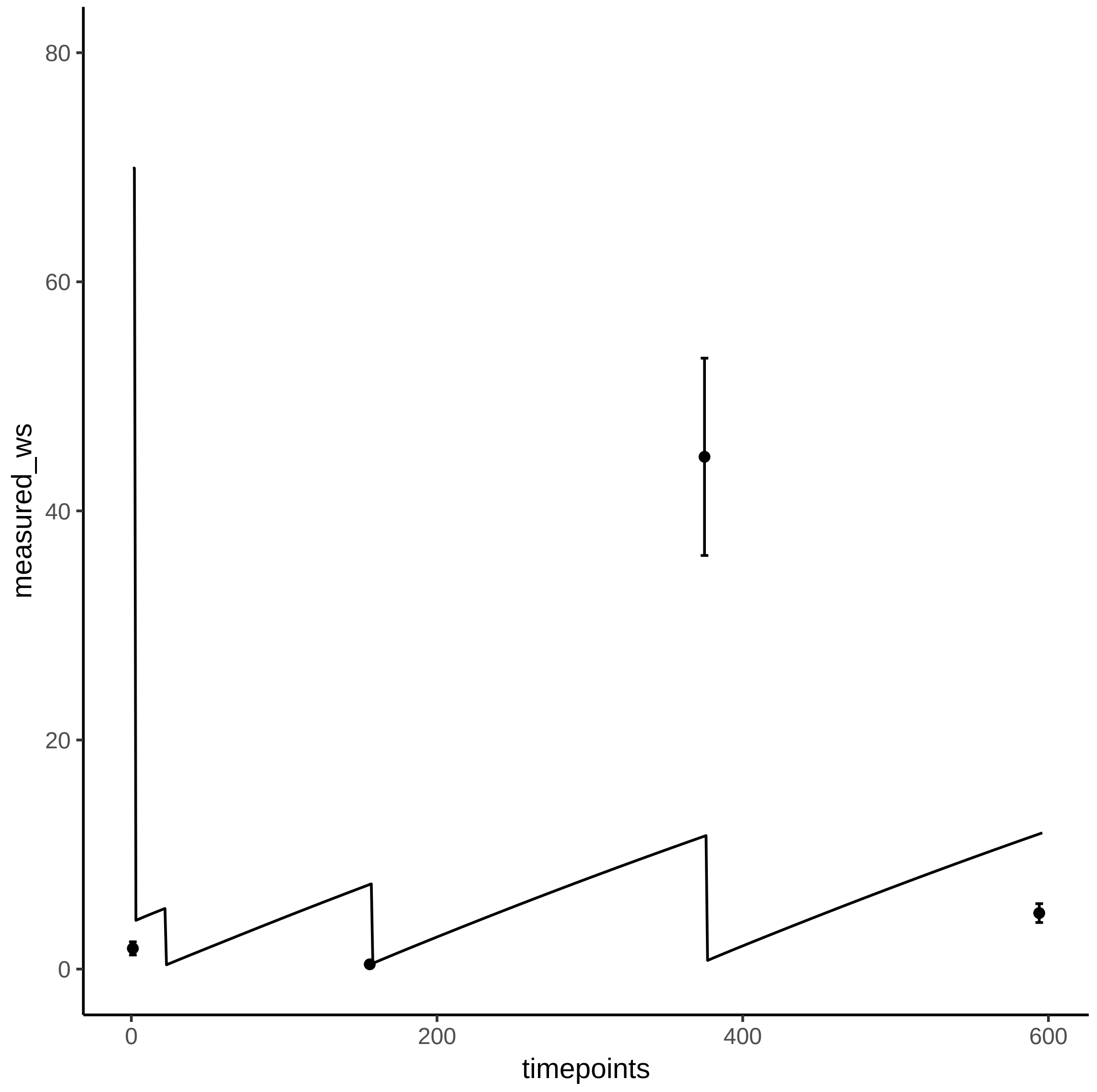


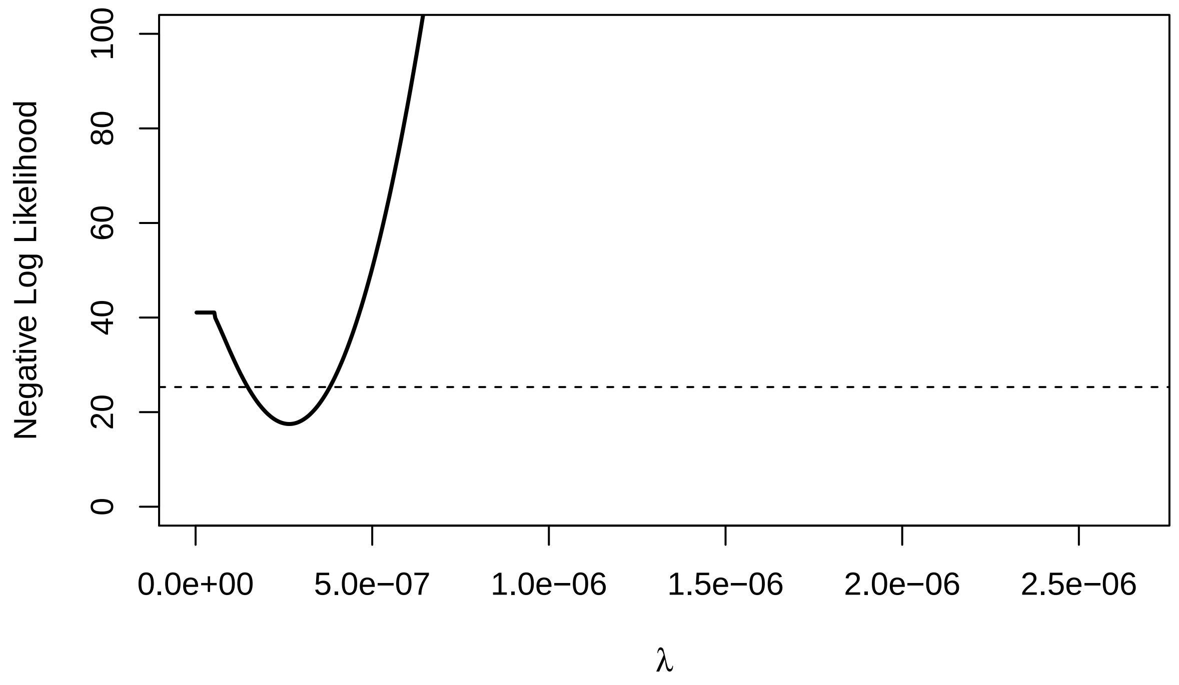
Figure 3: Single transmission parameter model fit to epi data resulting from best fit parameter set

```r
  epi_plot %>%
    ggplot(aes(x = timepoints, y = measured_ws)) +
      geom_point() +
      theme_classic() +
      ylim(0,80) +
      geom_errorbar(ymin = measured_ws - w_errors, ymax = measured_ws + w_errors, width = 5) +
      geom_line(data = fit_traj_1par, aes(x = time, y = Wt.cvrg))
```

So the main problem is the high transmission point, but we kind of knew that already. I think this works well, especially since it doesn't require summarizing the two seasonal transmission parameters into a single λ estimate to use in R_0 estimates. Let's next estimate the 95%CI from the profile likelihood as before and then get the range of R_0 resulting from this estimate

```r
#Get 95%CI boundary
  boundary_1par <- op_min_1par$objective + 7.815 #chi-square with 3 degree of freedom, 95% CI

#Grid search for 95%CI
  lambda_1par <- seq(bestLambda_1par/100, bestLambda_1par*10, length.out = 1000)

  negLL_1par <- sapply(lambda_1par, Tx_simulate_1par)
```

```
#plot begLL by lambda and include cutoff line that represents profile likelihood boundary
  plot(lambda_1par, negLL_1par, type = "l", lwd = 2, ylim = c(0,100),
       xlab = expression(lambda), ylab = "Negative Log Likelihood")
    abline(h = boundary_1par, lty = 2)
```



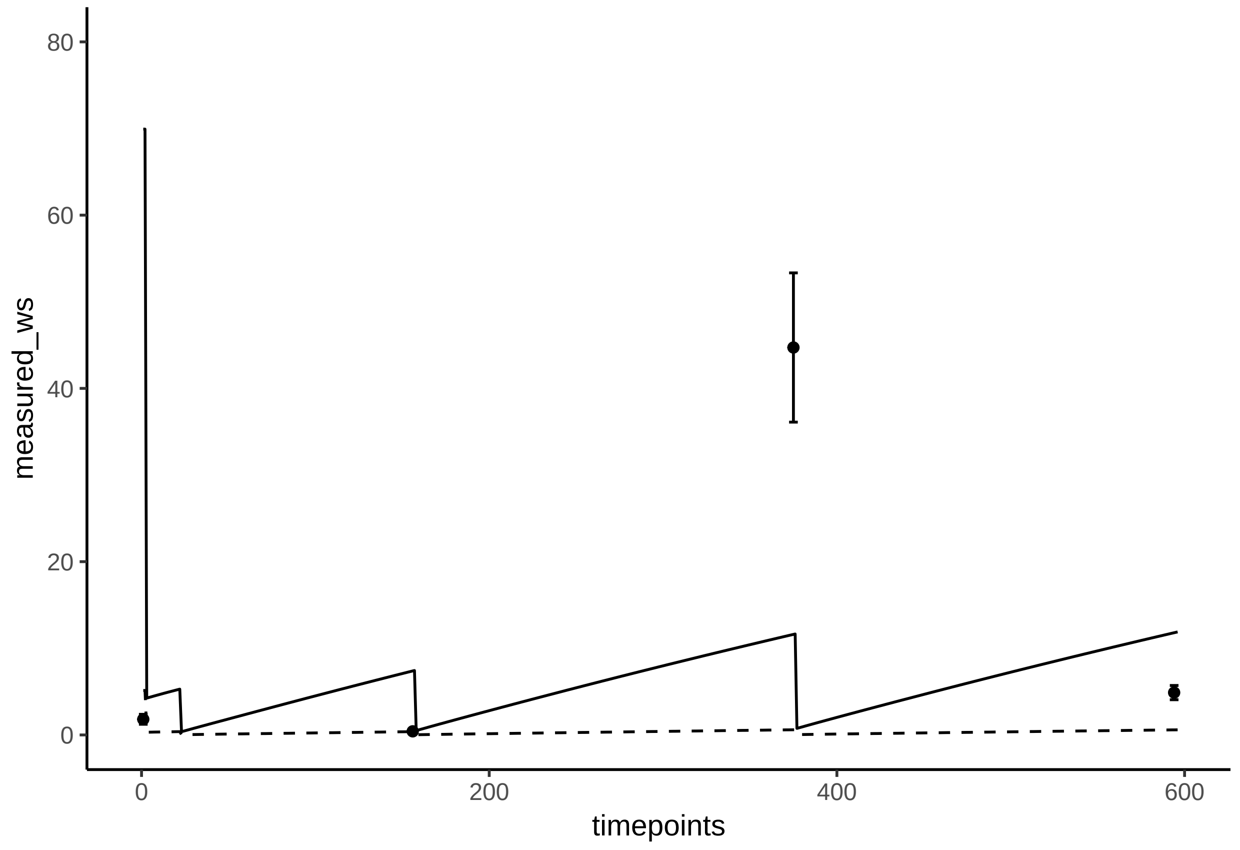
```
#generate matrix of parameters within 95%Ci, their fit, and an associated weight based on their fit, then save
  fit_1par_mat <- data.frame(cbind(lambda_1par, negLL_1par)) %>%
    filter(negLL_1par <= boundary_1par) %>%
    mutate(weight = 1/negLL_1par)

  save(fit_1par_mat, file = "../Models/trans_pars_fit.Rdata")
```

```
#Create parameter sets with these estimates to get R0 range
  #Point estimate
  point_r0_noPred <- r0.Ag(parameters = fit_pars_1par)[3]

  #lower limit
  fit_pars_1par_lo <- fit_pars_1par
    fit_pars_1par_lo["lambda"] <- min(fit_1par_mat$lambda_1par)
    lo_r0_noPred <- r0.Ag(parameters = fit_pars_1par_lo)[3]

  #Upper limit
  fit_pars_1par_hi <- fit_pars_1par
    fit_pars_1par_hi["lambda"] <- max(fit_1par_mat$lambda_1par)
    hi_r0_noPred <- r0.Ag(parameters = fit_pars_1par_hi)[3]

#Estimates with predators at reasonable wildlife density
  #Point estimate
  fit_pars_1par_pred <- fit_pars_1par
  fit_pars_1par_pred["mu_P"] <- init_pars["mu_P"]
    point_r0_Pred <- r0.Ag(parameters = fit_pars_1par_pred)[3]
```

```r
  #lower limit
  fit_pars_1par_lo_pred <- fit_pars_1par_lo
  fit_pars_1par_lo_pred["mu_P"] <- init_pars["mu_P"]
    lo_r0_Pred <- r0.Ag(parameters = fit_pars_1par_lo_pred)[3]

  #Upper limit
  fit_pars_1par_hi_pred <- fit_pars_1par_hi
  fit_pars_1par_hi_pred["mu_P"] <- init_pars["mu_P"]
    hi_r0_Pred <- r0.Ag(parameters = fit_pars_1par_hi_pred)[3]
```

Without predators, $R_0 = 2.7$ (95%CI: 2.02 - 3.23). With predators at natural densities, $R_0 = 2.05$ (95%CI: 1.53 - 2.45).

```r
#simulate with best fitting parameter
  fit_eq_vals_1par <- runsteady(y = base_mda.start, func = agrochem_mda_mod,
                                parms = fit_pars_1par)$y

  fit_traj_1par <- data.frame(ode(fit_eq_vals_1par, traj_time, agrochem_mda_mod,
                                  fit_pars_1par, events = list(data = mdas)))

#simulate with best fitting parameter and pred introduction
base_mda.preds = c(S = 40*area,
                   E = 0,
                   I = 0,
                   Wt = 2*cvrg,
                   Wu = 2*(1-cvrg),
                   P = 0.2*area)


  fit_eq_vals_1par_pred <- runsteady(y = base_mda.preds, func = agrochem_mda_mod,
                                     parms = fit_pars_1par_pred)$y

  fit_traj_1par_pred <- data.frame(ode(fit_eq_vals_1par_pred, traj_time, agrochem_mda_mod,
                                       fit_pars_1par_pred, events = list(data = mdas)))

  epi_plot %>%
    ggplot(aes(x = timepoints, y = measured_ws)) +
      geom_point() +
      theme_classic() +
      ylim(0,80) +
      geom_errorbar(ymin = measured_ws - w_errors, ymax = measured_ws + w_errors, width = 5) +
      geom_line(data = fit_traj_1par, aes(x = time, y = Wt.cvrg)) +
      geom_line(data = fit_traj_1par_pred, aes(x = time, y = Wt.cvrg), lty = 2)
```



Low and high transmisison season parameters

```
#what's the negLL if we just guess 0 for all data points? (Used as a point of comparison)
base_fit_bad <- -sum(mapply(pointLogLike, measured_ws, w_errors, c(0,0,0,0)))

#Function to simulate transmission, given transmission parameters and return negative log likelihood from fit to
Tx_simulate<-function(lambdas){

  params["lambda1"]<-lambdas[1]
  params["lambda2"]<-lambdas[2]
  params["lambda"]<-( (375/594) * lambdas[1]) +( (219/594) * lambdas[2])

    eqbm_vals <- runsteady(y = base_mda.start, func = agrochem_mda_mod_fit, parms = params)$y

    traj <- data.frame(ode(eqbm_vals, traj_time, agrochem_mda_mod_fit,
                           params, events = list(data = mdas)))

    modeled <- traj$Wt[timepoints + 2] * 0.5 * mapply(phi_Wk, traj$Wt[timepoints + 2], measured_ks)

    point_LL <- mapply(pointLogLike, measured_ws, w_errors, modeled)

    point_LL[!is.finite(point_LL)] <- NA

    logLike <- sum(point_LL, na.rm = TRUE)

    return(-logLike)

}

#Fit using Optim
  op_min <- optim(par=c(bestLambda_1par, bestLambda_1par*10),
                 Tx_simulate, method="Nelder-Mead")

#Extract best fitting parameters
  bestLambda1<-op_min$par[1]
  bestLambda2<-op_min$par[2]

  fit_pars <- init_pars

  fit_pars["lambda1"]<-bestLambda1
  fit_pars["lambda2"]<-bestLambda2
  fit_pars["lambda"] <- (375/594) * bestLambda1 + (219/594) * bestLambda2

#simulate with best fitting parameters
  fit_eq_vals <- runsteady(y = base_mda.start, func = agrochem_mda_mod_fit, parms = fit_pars)$y

    fit_traj <- data.frame(ode(fit_eq_vals, traj_time, agrochem_mda_mod_fit,
                           fit_pars, events = list(data = mdas)))

  epi_plot %>%
    ggplot(aes(x = timepoints, y = measured_ws)) +
      geom_point() +
      theme_classic() +
      ylim(0,80) +
      geom_errorbar(ymin = measured_ws - w_errors, ymax = measured_ws + w_errors, width = 5) +
      geom_line(data = fit_traj, aes(x = time, y = Wt.cvrg))
```

Fit doesn't look too bad actually. Now find all parameter sets that lie within the 95% CI

```
#Range of low and high transmission season lambdas to test
  lambda_los <- seq(init_pars["lambda"]/100, init_pars["lambda"]*10, length.out = 50)
  lambda_his <- seq(init_pars["lambda"]/10, init_pars["lambda"]*100, length.out = 50)

  lambda_tests <- expand.grid(lambda_los = lambda_los, lambda_his = lambda_his, negLL = 0)

  params <- init_pars
  params["lambda1"] <- lambda_los[25]
  params["lambda2"] <- lambda_his[25]

#test all parameter combinations to find best starting place for optimization
  lambda_tests$negLL <- apply(lambda_tests, 1, Tx_simulate)


#Value of negLL that serves as cutoff for 95%CI
  boundary <- op_min$value + 7.815 #chi-square with 3 degree of freedom, 95% CI

#Keep obs within 95%CI
  lambda_tests95ci <- lambda_tests %>%
    filter(negLL <= boundary)

#Plot heatmap with 95%CI delineated
lambda_tests %>%
  ggplot(aes(x = lambda_los, y = lambda_his, fill = negLL, z = negLL)) +
    geom_tile() +
    geom_contour(breaks = c(boundary), col = "black") +
    scale_fill_viridis(option = "magma", direction = -1) +
    labs(y = expression(lambda[hi]),
         x = expression(lambda[lo])) +
    theme_bw() +
    theme(axis.ticks=element_blank(),
```

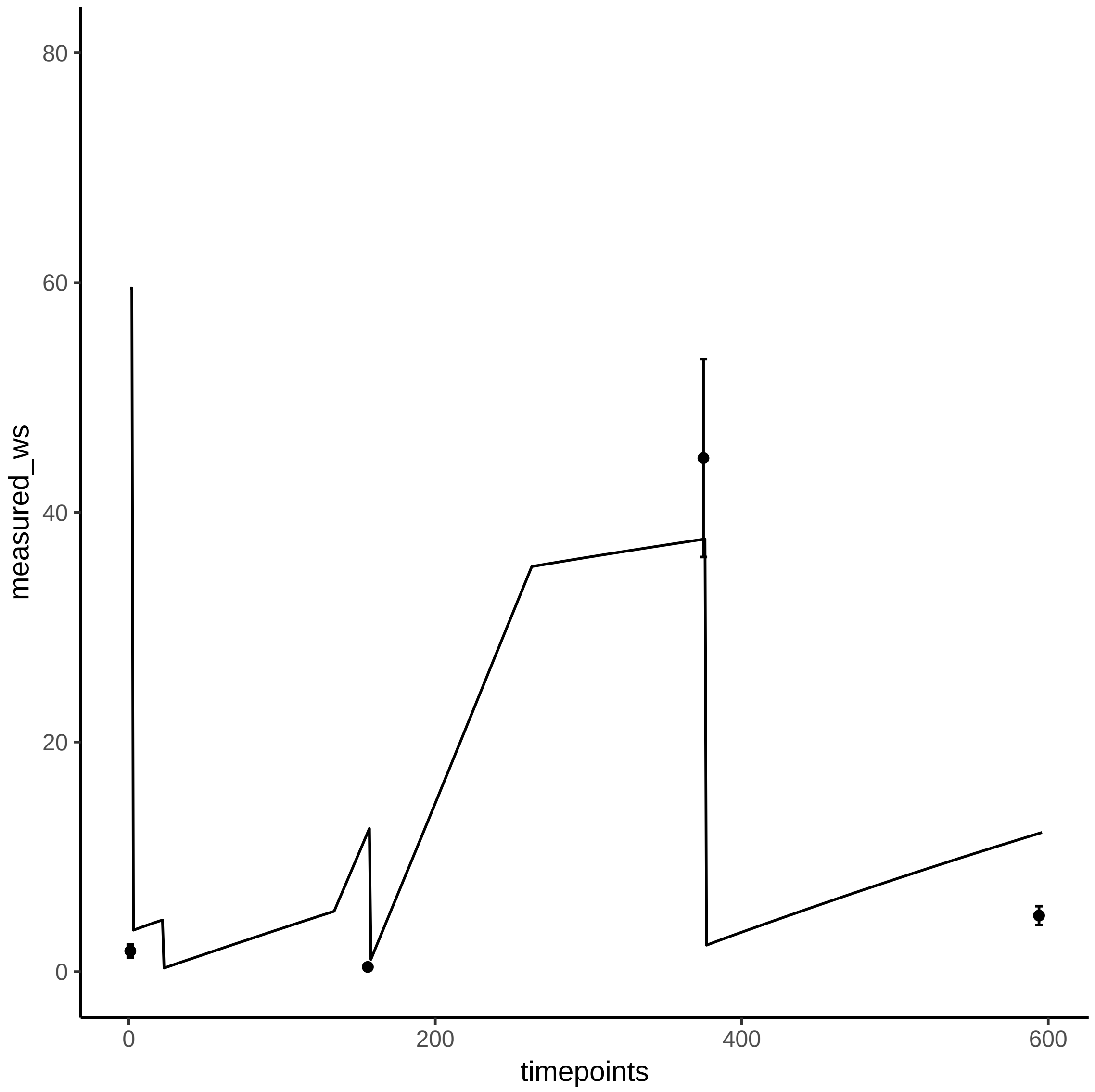


Figure 4: Two transmission parameter model fit to epi data resulting from best fit parameter set

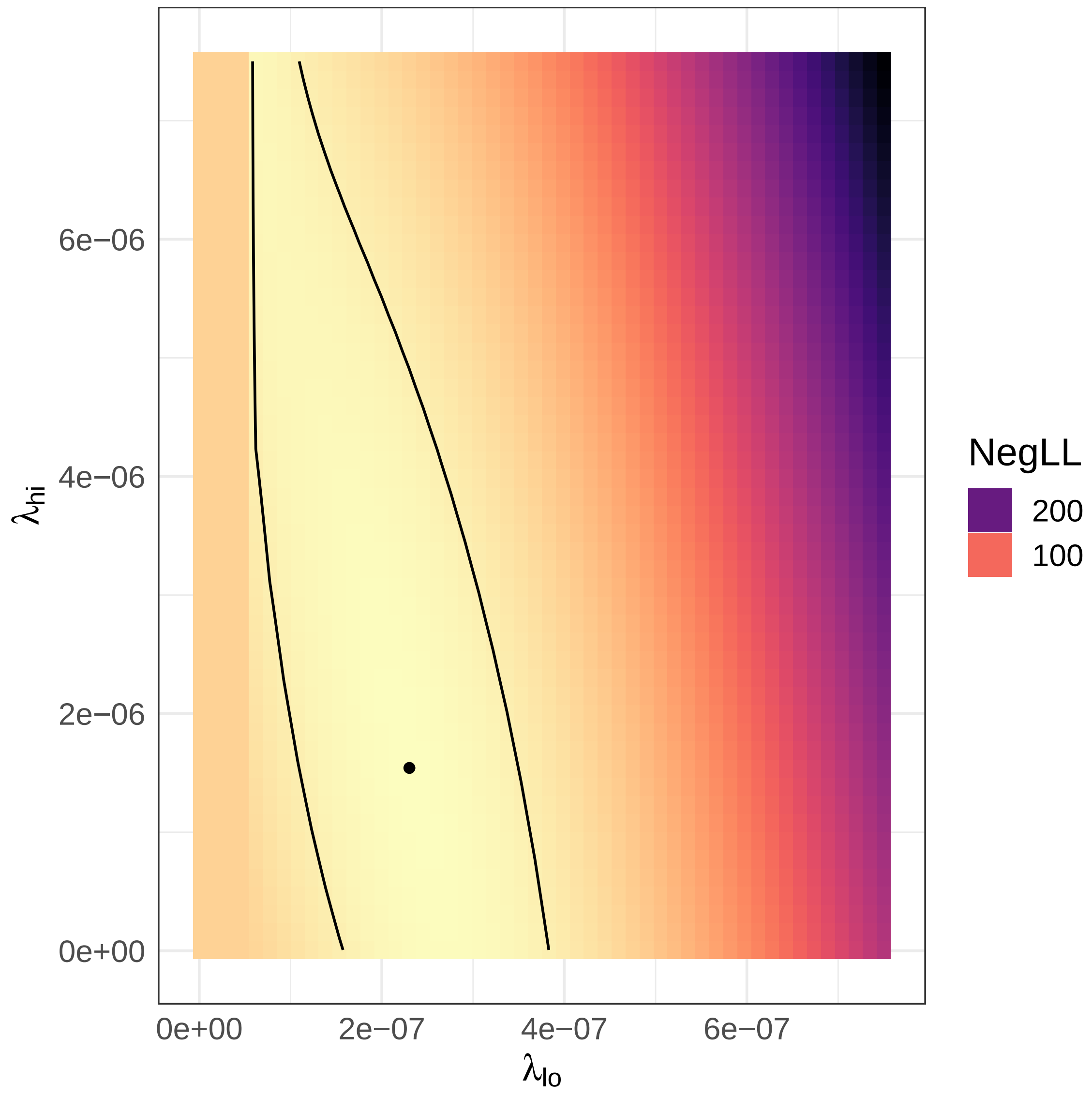


Figure 5: Heat map of transmission parameters and the negative log likelihood resulting form their fit to epi data. Black contour indicates 95% CI region and black point indicates coordinates of best fit parameters

```
          axis.text=element_text(size=12),
          axis.title=element_text(size=14),
          legend.title=element_text(size=15),
          legend.text=element_text(size=12)) +
    guides(fill=guide_legend(title="NegLL",
                             reverse = T)) +
    annotate(geom = "point", x = bestLambda1, y = bestLambda2, col = "black")
```

This kind of makes it look like the likelihood profile doesn't really have a bottom to it, i.e. given the right low transmission season parameter, high transmission season is unnecessary. That makes sense given the fitting above with one transmission parameter works fine, but let's check that out formally

```
Tx_simulate(c(bestLambda1, 0))
```

```
## [1] 18.6806
```

```
boundary
```

```
## [1] 24.76172
```

Indeed, high transmission parameter appears unneccessary. However it also appears that the range of Lambdas tested above doesn't encapsulate the full range of parameter values that could feasibly lie within the 95%CI, so let's expand the grid, but only in the dimensions that appear reasonable based on the above. In particular, it appears that we need to explore a broader range of λ_{hi} values, but can afford to restrict the $\lambda_{l}o$ values a bit.

```
#Range of low and high transmission season lambdas to test
  lambda_los2 <- seq(init_pars["lambda"]/100, init_pars["lambda"]*8, length.out = 30)
  lambda_his2 <- seq(0, init_pars["lambda"]*150, length.out = 30)

  lambda_tests2 <- expand.grid(lambda_los = lambda_los2, lambda_his = lambda_his2, negLL = 0)

  params <- init_pars
  params["lambda1"] <- lambda_los2[15]
  params["lambda2"] <- lambda_his2[15]

#test all parameter combinations and get model fits
  lambda_tests2$negLL <- apply(lambda_tests2, 1, Tx_simulate)

#Keep obs within 95%CI
  lambda_tests2_95ci <- lambda_tests2 %>%
    filter(negLL <= boundary)

#Plot heatmap with 95%CI delineated
lambda_tests2 %>%
  ggplot(aes(x = lambda_los, y = lambda_his, fill = negLL, z = negLL)) +
    geom_tile() +
    geom_contour(breaks = c(boundary), col = "black") +
    scale_fill_viridis(option = "magma", direction = -1) +
    labs(y = expression(lambda[hi]),
         x = expression(lambda[lo])) +
    theme_bw() +
    theme(axis.ticks=element_blank(),
          axis.text=element_text(size=12),
          axis.title=element_text(size=14),
          legend.title=element_text(size=15),
          legend.text=element_text(size=12)) +
    guides(fill=guide_legend(title="NegLL",
                             reverse = T)) +
    annotate(geom = "point", x = bestLambda1, y = bestLambda2, col = "black")
```

So that appears to be true... Adding the second transmission parameter corresponding to high transmission season only improves the model fit marginally (negative log likelihood from 17.49 in the single parameter model as opposed to 16.95 in the model with two parameters). So definitely prefer the simpler, single transmission parameter at this point, but let's try the seasonal model as well

Seasonality model

```
#Function to simulate seasonal transmission and return negative log likelihood from fit to epi data ###########
Tx_simulate_seasonal<-function(lambda){

  params["lambda"] <- lambda

    eqbm_vals <- runsteady(y = base_mda.start, func = agrochem_mda_mod, parms = params)$y

    traj <- data.frame(ode(eqbm_vals, traj_time, agrochem_mda_seasonal_mod_fit,
                           params, events = list(data = mdas)))

    modeled <- traj$Wt[timepoints + 2] * 0.5 * mapply(phi_Wk, traj$Wt[timepoints + 2], measured_ks)
```

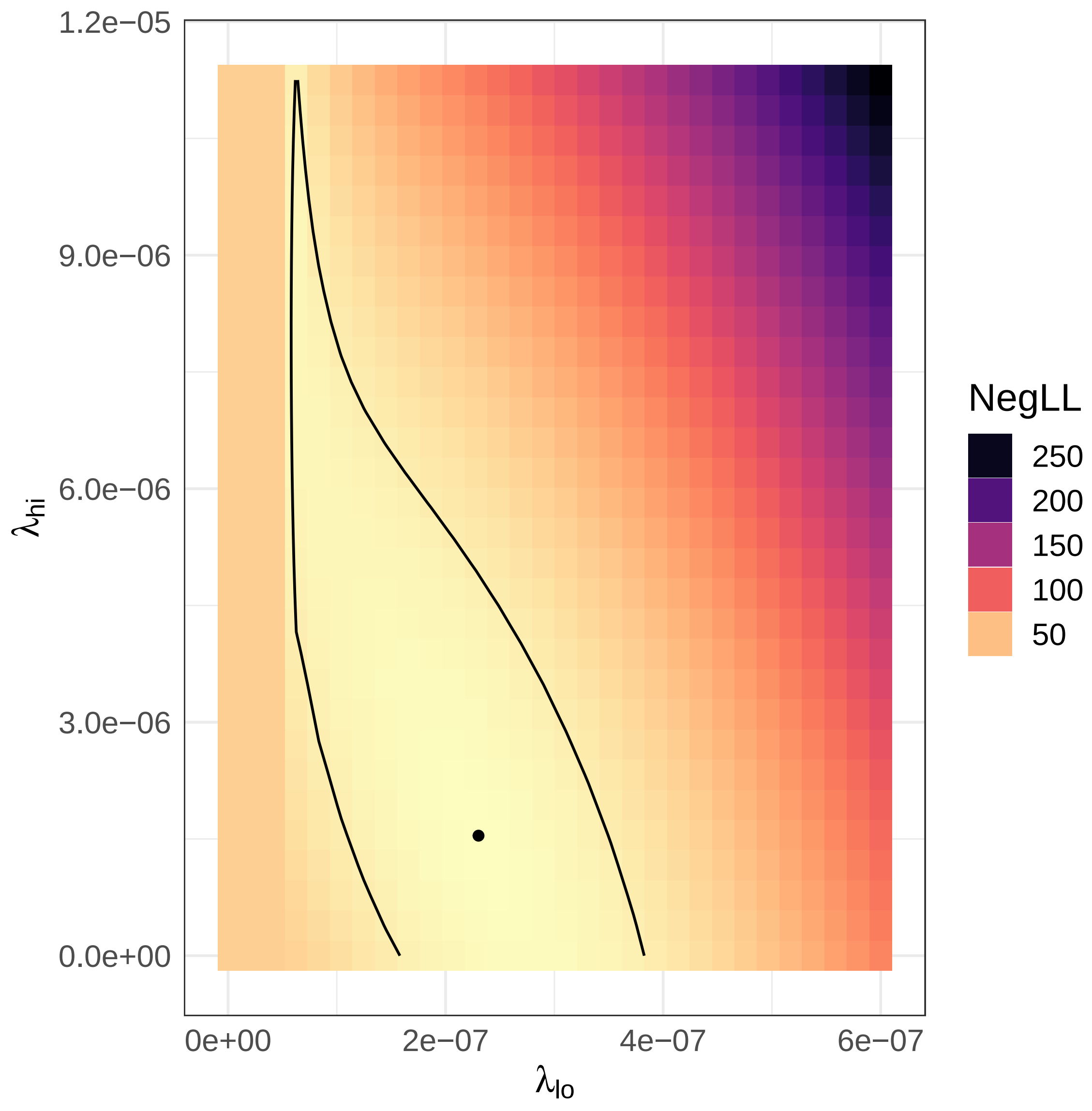
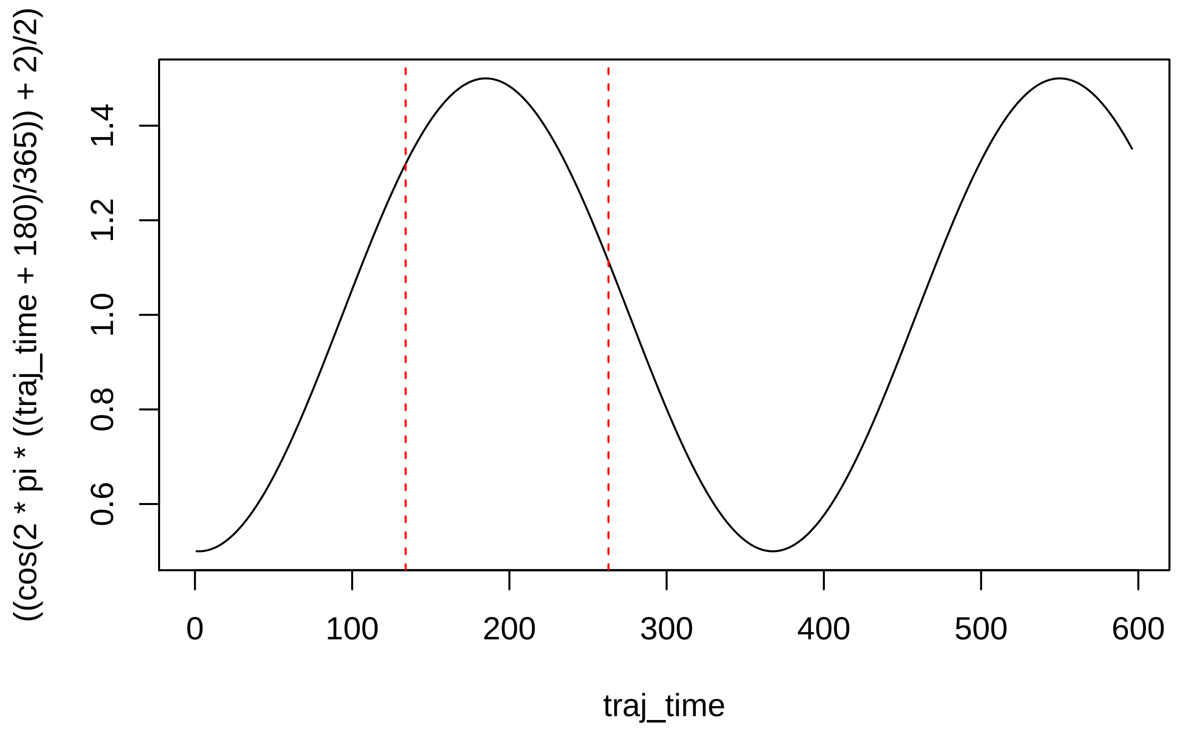
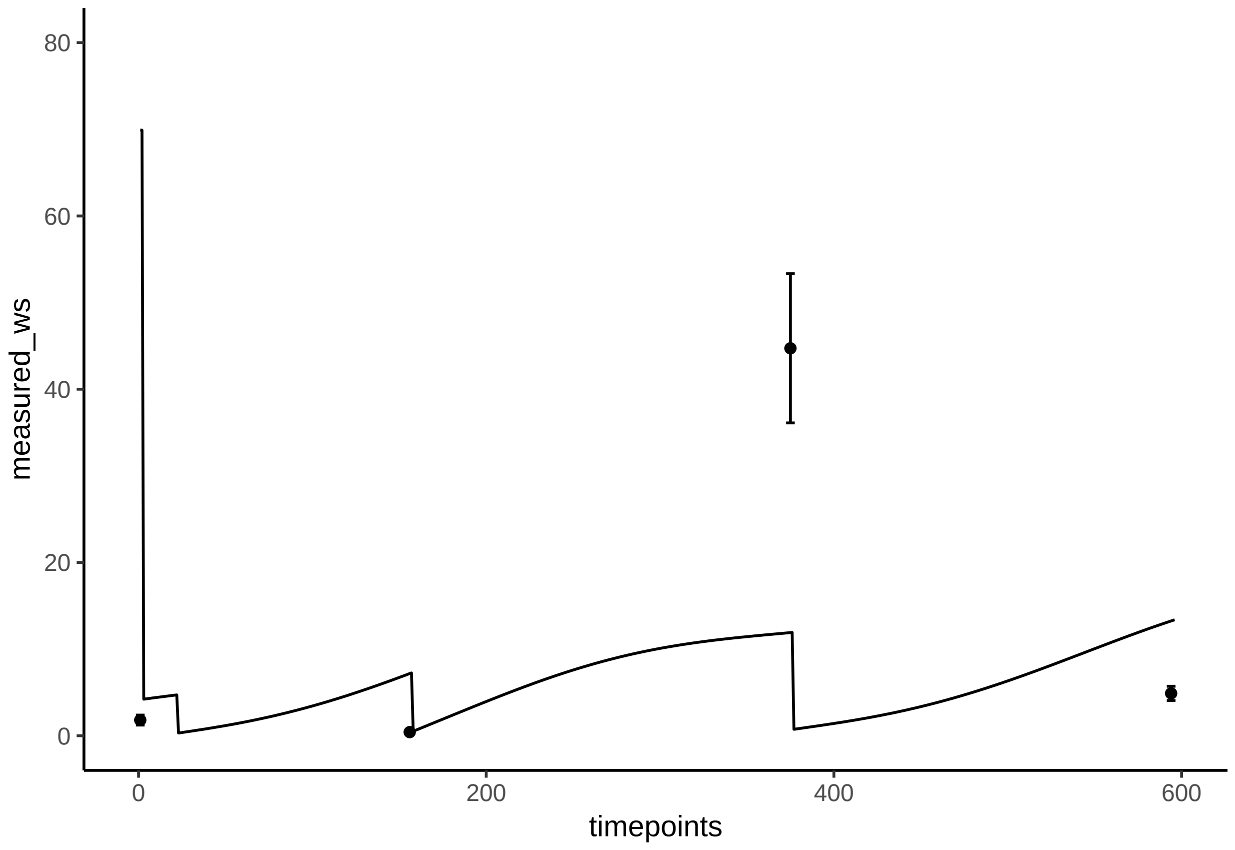


Figure 6: Heat map of expanded range of transmission parameters and the negative log likelihood resulting form their fit to epi data. Black contour indicates 95% CI region and black point indicates coordinates of best fit parameters

```r
    point_LL <- mapply(pointLogLike, measured_ws, w_errors, modeled)

    point_LL[!is.finite(point_LL)] <- NA

    logLike <- sum(point_LL, na.rm = TRUE)

    return(-logLike)
}

#First make sure that seasonality matches up with transmission seasons
plot(traj_time, ((cos(2*pi*((traj_time+180)/365)) + 2)/2), type = "l")
  abline(v = c(seasons[c(2,3)]), lty = 2, col = 2)
```



```r
#Fit using Optimize for 1-dimensional optimization
  op_min_seasonal <- optimize(Tx_simulate_seasonal, interval = c(bestLambda_1par/100, bestLambda_1par*10))

#Extract best fitting parameter
  bestLambda_seasonal <- op_min_seasonal$minimum

  fit_pars_1par <- init_pars
  fit_pars_1par["lambda"] <- bestLambda_1par
  fit_pars_1par["mu_P"] <- 0.9  # Want to estimate predator-free R0

#simulate with best fitting parameter
  fit_eq_vals_seasonal <- runsteady(y = base_mda.start, func = agrochem_mda_mod,
                                    parms = fit_pars_1par)$y

  fit_traj_seasonal <- data.frame(ode(fit_eq_vals_1par, traj_time, agrochem_mda_seasonal_mod_fit,
                                      fit_pars_1par, events = list(data = mdas)))

  epi_plot %>%
    ggplot(aes(x = timepoints, y = measured_ws)) +
      geom_point() +
      theme_classic() +
      ylim(0,80) +
      geom_errorbar(ymin = measured_ws - w_errors, ymax = measured_ws + w_errors, width = 5) +
      geom_line(data = fit_traj_seasonal, aes(x = time, y = Wt.cvrg))
```


measured_ws
timepoints