Assignment 6: Hunting Assignment

Taylor Cook

2025-05-12

Part 1

Build this model (e.g add hunting to the lotvmodK.R),

Some requirements/hints for your model

You should make sure that you don't hunt more prey than exist.

To ensure that you might also add a minimum prey population input that must be met before hunting is allowed.

Note you can make this as simple or as complex as you would like. You could represent hunting in a way that is similar to "harvesting" in the last assignment.

```
source("../R/predpreyhunt.R")
lotvmodK_hunt
```

```
## function (t, pop, pars)
## {
##     with(as.list(c(pop, pars)), {
##         dprey = rprey * (1 - prey/K) * prey - alpha * prey *
##             pred
##         if (prey >= Th) {
##             dprey = rprey * (1 - prey/K) * prey - alpha * prey *
##                 pred - rhunt * prey
##         }
##         dpred = eff * alpha * prey * pred - pmort * pred
##         return(list(c(dprey, dpred)))
##     })
## }
```

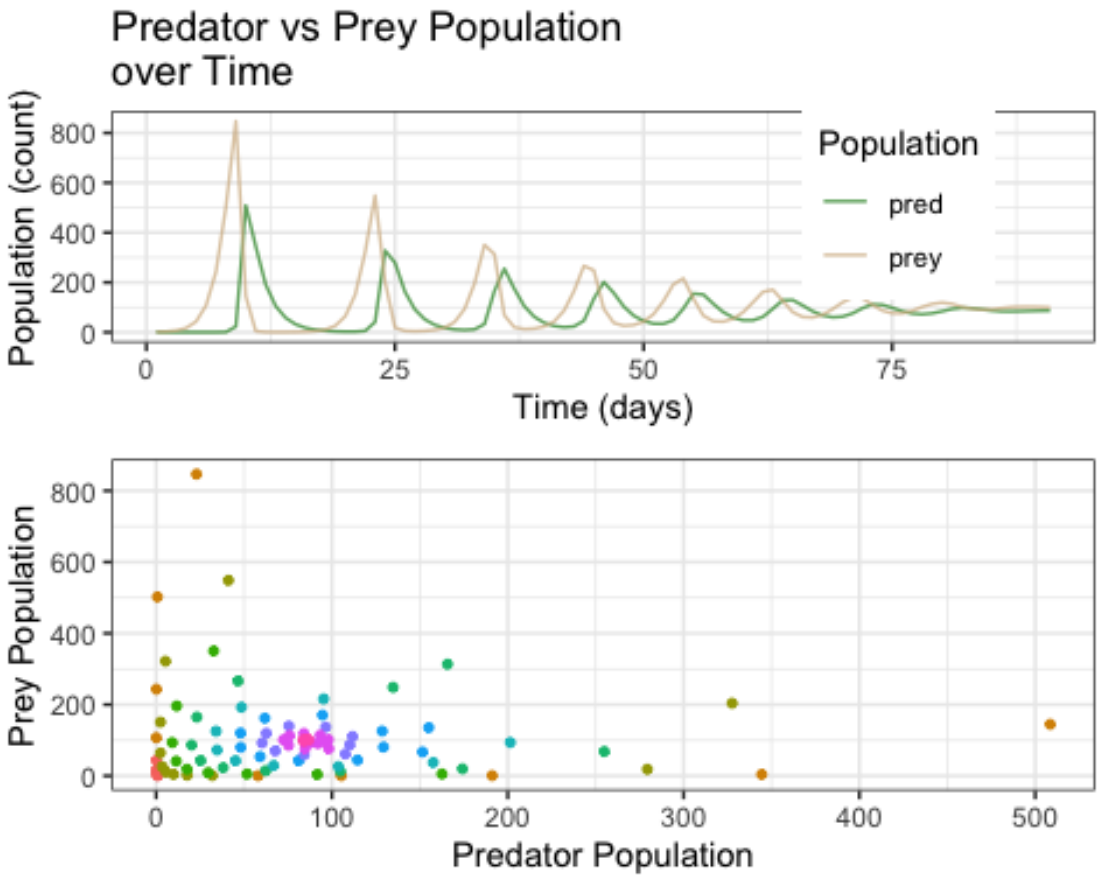
```
currpop=c(prey=1,pred=1)

pars = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.05,Th=100)

days=seq(from=1,to=300)

res = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars)
```

```
## DLSODA-  At current T (=R1), MXSTEP (=I1) steps
##       taken on this call before reaching TOUT
## In above message, I1 = 5000
##
## In above message, R1 = 90.8925
##

## Warning in lsoda(y, times, func, parms, ...): an excessive amount of work
(>
## maxsteps ) was done, but integration was not successful - increase
maxsteps

## Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
## accurate, as far as they go
```

```
resl = as.data.frame(res) %>%
  pivot_longer(-time, names_to = "species", values_to = "population")

plot <- ggplot(resl, aes(x = time, y = population, color = species)) +
  geom_line(alpha = 0.7) +
  theme_bw() +
  scale_color_manual(values = c("forestgreen", "tan")) +
  labs(x = "Time (days)", y = "Population (count)", color = "Population") +
  ggtitle("Predator vs Prey Population\nover Time") +
  theme(legend.position = c(0.8, 0.6))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated
in ggplot2
## 3.5.0.
## ℹ Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
plot2 <- ggplot(as.data.frame(res), aes(x = pred, y = prey,
col=as.factor(round(time/10)))) +
  geom_point(size = 1) +
  theme_bw() +
  labs(x = "Predator Population", y = "Prey Population") +
  theme(legend.position = "none")

plot_grid(plot, plot2, nrow = 2)
```



Part 2

Explore how different hunting levels and different minimum prey populations (before hunting is allowed) are likely to effect the stability of the populations of both predator and prey. A key challenge is how you might want to define stability? It is up to you but you will need to write a sentence to explain why you chose the measure that you did. It could be something as simple as maintaining a population above some value 50 years into the future.

Use this exploration to recommend a hunting target that will be sustainable (e.g leave you with a stable prey and predator population).

It is up to you how you "explore" hunting - you can simply try different values of the parameters in your hunting model or do it more formally by running your model across a range of values. You could think about parameter interactions

You can assume the following are best guesses of key parameters

rprey=0.95, alpha=0.01, eff=0.6,pmort=0.4, K=2000,

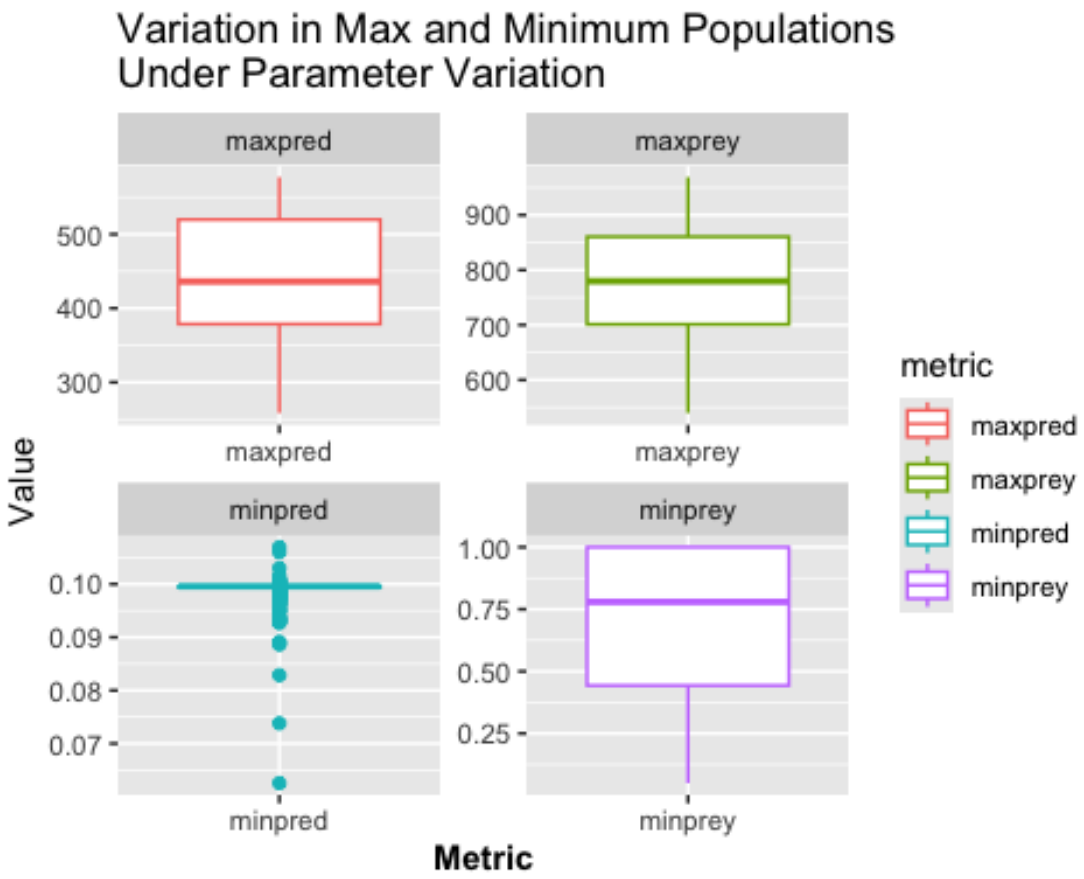
Defining Stability:

I defined stability as maintaining some stable population of individuals for both the predator and prey population. In this scenario, hunting played a part in reducing the prey population up until about the 35th day, where then the prey population was reduced below the minimum threshold, and hunting was no longer allowed. In this scenario, the populations stabilized below the minimum threshold, which would indicate that hunting may not be allowed to continue in order to maintain stable populations in this setting.

```
## Creating 2 new sets of parameters, then going to run the function over
each of them and bind dataframes to see differences in how rhunt affects
predator and prey populations over time

## Setting first set of parameter values
rhunt = rnorm(200, 0.1, 0.083)
Th = rnorm(200, 100, 33)

## putting them into a dataframe
parms1 = cbind.data.frame(rhunt = rhunt, Th = Th)

## Setting first set of parameter values
rhunt = rnorm(200, 0.1, 0.083)
Th = rnorm(200, 100, 33)

## putting them into a dataframe
parms2 = cbind.data.frame(rhunt = rhunt, Th = Th)

## Creating sensitivity object of parameter sets
sens_PP = sobolSalt(model = NULL,parms1, parms2, nboot = 300)

## Setting names
colnames(sens_PP$X) = c("rhunt", "Th")

## Creating metrics function to calculate variation in extremes of
populations from the ode solver
compute_metrics = function(result) {
  maxprey = max(result$prey)
  maxpred = max(result$pred)
  minprey = min(result$prey)
  minpred = min(result$pred)
  return(list(maxprey=maxprey, minprey=minprey, maxpred=maxpred,
minpred=minpred))
  }

## Creating wrapper function to write the lotvmodK_hunt function into ode
solver and use compute_metrics function to calculate maximum and minimum
variation for either population under varying rates of hunting (rhunt) and
minimum thresholds for hunting to be allowed (Th)
```

```r
wrapper_func <- function(rprey, alpha, eff, pmort, K, rhunt, Th, currpop,
days, func) {
  parms = list(rprey=rprey, alpha=alpha, eff=eff, pmort=pmort, K=K,
rhunt=rhunt, Th=Th)
  result = ode(y=currpop, times=days, func=func, parms=parms)
  colnames(result) = c("year", "prey", "pred")

  metrics = compute_metrics(as.data.frame(result))
  return(metrics)
}

## Setting current population
currpop = c(prey=1, pred=1)

## Setting days (years) to sequence over
days=seq(from=1,to=500)

## Using pmap to map my sensitivity parameter values to the wrapper function
and see how maximum and minimum populations vary over variation in rhunt and
Th
calculation = as.data.frame(sens_PP$X) %>%
  pmap(wrapper_func,
       rprey=0.95,
       alpha=0.01,
       eff=0.6,
       pmort=0.6,
       K=2000,
       currpop=currpop,
       days=days,
       func=lotvmodK_hunt)

## Setting results as data frame
results = as.data.frame(calculation)

## Setting column names
results = calculation %>%
map_dfr(`[`,c("maxprey","minprey","maxpred","minpred"))

## pivoting longer to be able to graph metrics
resultsl = results %>%
  pivot_longer(cols=c(1:4), names_to = "metric", values_to = "value")

## Graphing metrics under variation
ggplot(resultsl, aes(x = metric, y = value, color = metric)) +
  geom_boxplot() +
  labs(x = "Metric", y = "Value") +
  ggtitle("Variation in Max and Minimum Populations\nUnder Parameter
Variation") +
```

```
  facet_wrap(~metric, scales = "free") +
  theme(axis.title.x = element_text(face = "bold"))
```

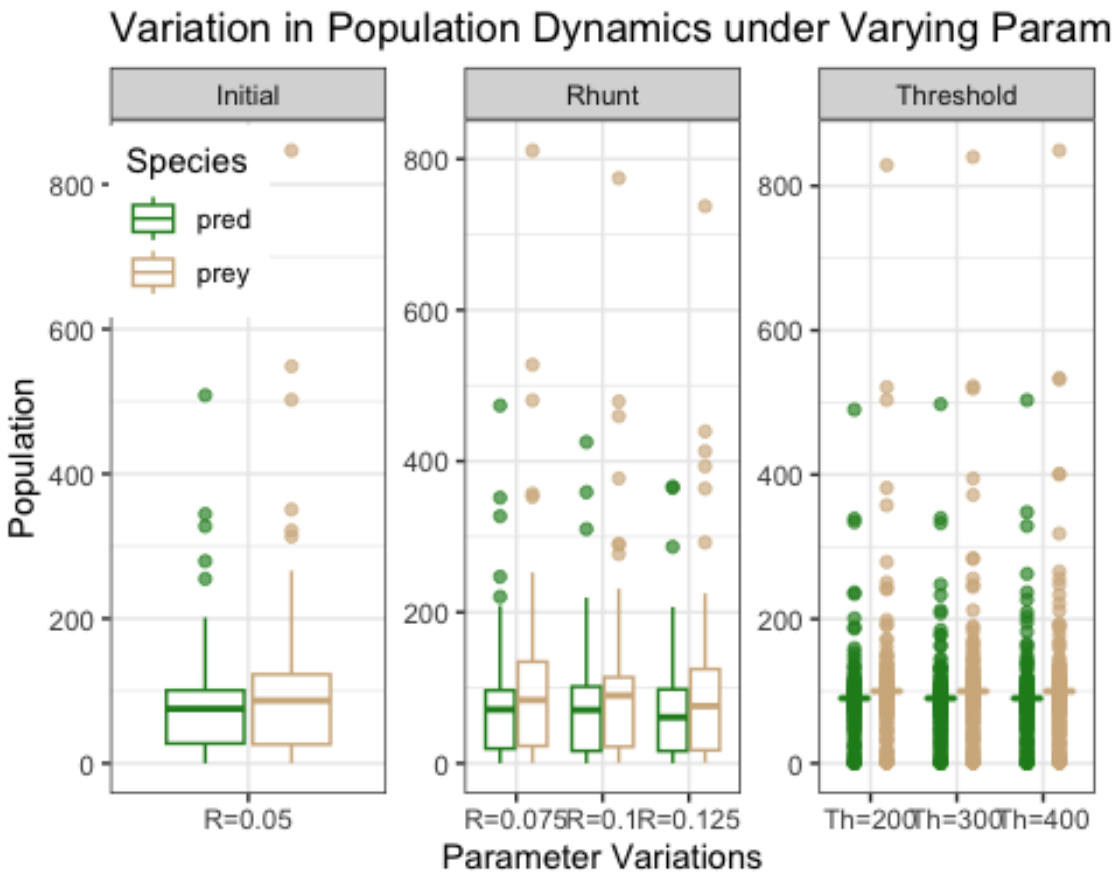



```
################ Variation in hunting rate, constant threshold
################
 ## Rhunt = 0.1
 pars1 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.075,Th=100)
 res1 = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars1)

## DLSODA-  At current T (=R1), MXSTEP (=I1) steps
##       taken on this call before reaching TOUT
## In above message, I1 = 5000
##
## In above message, R1 = 76.772
##

 res11 = as.data.frame(res1) %>%
   pivot_longer(-time, names_to = "species", values_to = "population1") %>%
   pivot_longer(population1, names_to = "iteration", values_to =
"population")

 ## Rhunt = 0.15
 pars2 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
```

```
rhunt=0.1,Th=100)
 res2 = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars2)

## DLSODA-  At current T (=R1), MXSTEP (=I1) steps
##       taken on this call before reaching TOUT
## In above message, I1 = 5000
##
## In above message, R1 = 67.244
##

 resl2 = as.data.frame(res2) %>%
   pivot_longer(-time, names_to = "species", values_to = "population2") %>%
   pivot_longer(population2, names_to = "iteration", values_to =
"population")

 ## Rhunt = 0.203 because ode solver was being weird at 0.20
 pars3 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.125,Th=100)
 res3 = deSolve::ode(func=lotvmodK_hunt, y = currpop, times = days, parms =
pars3)

## DLSODA-  At current T (=R1), MXSTEP (=I1) steps
##       taken on this call before reaching TOUT
## In above message, I1 = 5000
##
## In above message, R1 = 57.7917
##

 resl3 = as.data.frame(res3) %>%
   pivot_longer(-time, names_to = "species", values_to = "population3") %>%
   pivot_longer(population3, names_to = "iteration", values_to =
"population")

################ Variation in threshold, constant hunting rate
################
 ## Threshold = 200
 pars4 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.075,Th=200)
 res4 = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars4)

 resl4 = as.data.frame(res4) %>%
   pivot_longer(-time, names_to = "species", values_to = "population4") %>%
   pivot_longer(population4, names_to = "iteration", values_to =
"population")
 ## Threshold = 300
 pars5 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.075,Th=300)
 res5 = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars5)

 resl5 = as.data.frame(res5) %>%
```

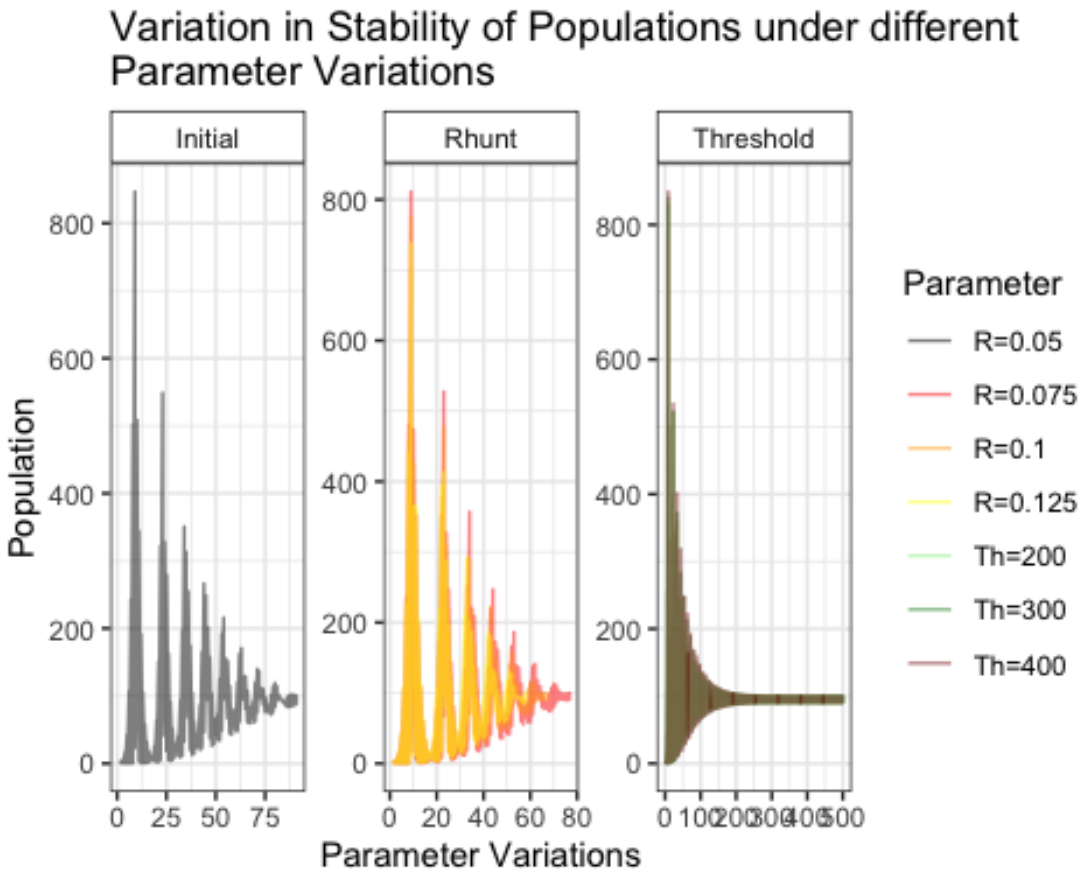
```r
   pivot_longer(-time, names_to = "species", values_to = "population5") %>%
   pivot_longer(population5, names_to = "iteration", values_to =
"population")

## Threshold = 400
pars6 = c(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.075,Th=400)
res6 = ode(func=lotvmodK_hunt, y = currpop, times = days, parms = pars6)

 resl6 = as.data.frame(res6) %>%
   pivot_longer(-time, names_to = "species", values_to = "population6") %>%
   pivot_longer(population6, names_to = "iteration", values_to =
"population")

## Binding the dataframes together to be able to plot them all with their
different populations

 res_all <- resl %>% pivot_longer(population, names_to = "iteration",
values_to = "population") %>%
   rbind(., resl1) %>%
   rbind(., resl2) %>%
   rbind(., resl3) %>%
   rbind(., resl4) %>%
   rbind(., resl5) %>%
   rbind(., resl6) %>%
   mutate(parameter = case_when(
     iteration == "population" ~ "Initial",
     iteration == "population1" ~ "Rhunt",
     iteration == "population2" ~ "Rhunt",
     iteration == "population3" ~ "Rhunt",
     iteration == "population4" ~ "Threshold",
     iteration == "population5" ~ "Threshold",
     iteration == "population6" ~ "Threshold"
   ),
   label = case_when(
     iteration == "population" ~ "R=0.05",
     iteration == "population1" ~ "R=0.075",
     iteration == "population2" ~ "R=0.1",
     iteration == "population3" ~ "R=0.125",
     iteration == "population4" ~ "Th=200",
     iteration == "population5" ~ "Th=300",
     iteration == "population6" ~ "Th=400"
   ))

## Plotting population with varied parameters
ggplot(res_all, aes(x = as.factor(label), y = population, color = species)) +
  geom_boxplot(alpha = 0.65) +
  theme_bw() +
  scale_color_manual(values = c("forestgreen", "tan")) +
```

```
  labs(x = "Parameter Variations", y = "Population", color = "Species") +
  ggtitle("Variation in Population Dynamics under Varying Parameters")+
  facet_wrap(~parameter, scales = "free") +
  theme(legend.position = c(0.08, 0.85))
```



Plotting Stability Parameters Overtime

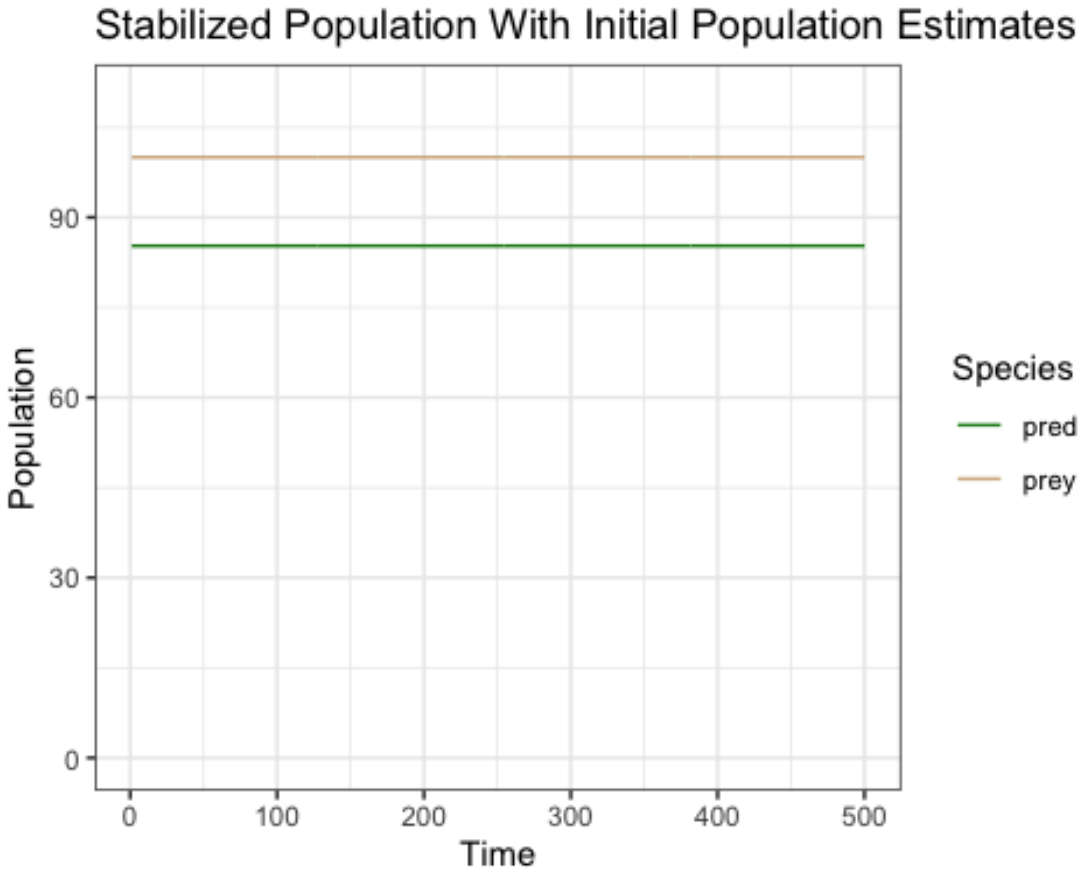
```
 ggplot(res_all, aes(x = time, y = population, color = label)) +
  geom_line(alpha = 0.5) +
  theme_bw() +
  scale_color_manual(values = c("black", "red", "orange", "yellow",
"lightgreen", "forestgreen", "brown4")) +
  labs(x = "Parameter Variations", y = "Population", color = "Parameter") +
  ggtitle("Variation in Stability of Populations under different\nParameter
Variations") +
  facet_wrap(~parameter, scales="free") +
  theme(strip.background = element_rect(color = "black", fill = "white"))
```



Defining Stability

```
# rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000, rhunt=0.25,Th=300
parms = data.frame(rprey=0.95, alpha=0.01, eff=0.6,pmort=0.6, K=2000,
rhunt=0.05, Th=100)

preyi = with(parms, pmort/(eff*alpha))
## 100 individuals

predi = with(parms, (rprey/alpha*(1-preyi/K) - rhunt/alpha))
## 85.25 individuals

## Setting current population to established starting values
currpop=c(prey=preyi, pred = predi)

## Setting days to sequence
days = seq(from=1,to=500,by=1)

## Running ode sovler to see results
results = ode(func=lotvmodK_hunt, y = currpop, times=days, parms = parms)

## Pivoting to plot
resultsl = as.data.frame(results) %>%
```

```
  pivot_longer(-time, names_to = "species", values_to = "population")

## Plotting Stable populations
ggplot(resultsl, aes(x =time, y = population, color = species)) +
  geom_line() +
  theme_bw() +
  labs(x = "Time", y = "Population", color = "Species") +
  ggtitle("Stabilized Population With Initial Population Estimates") +
  scale_color_manual(values = c("forestgreen", "tan")) +
  ylim(0,110)
```



From this analysis, it appears that stability can be obtained through two methods: The first method is by limiting the hunting rate to 7.5% of the prey population, as this achieves 'stability' the fastest (i.e. the population stabilizes faster below varying thresholds with hunting at 7.5% of the prey population, than when hunting varies at a constant minimum prey population threshold of 100). The second method would be to wait untli the prey and predator populations are 100 and 85, respectively, before allowing hunting of 5% of the prey population once it reaches a minimum thresthold of 300. While there are multiple ways to achieve stability, without knowing more about the population dynamics and varying other parameters, it is difficult to determine an optimum hunting rate other than between 5% and 10%. Therefore, we would recommend a conservative approach of 5-

7.5% hunting rate to allow for variations in population dynamics and stll maintain adequate levels of either population