Lab 3

Sawyer Balint

Fall 2024; Marine Semester Block 3

Contents

1 Introduction

This document is available at https://github.com/sjbalint/BI521/tree/main/scripts/labs

```
#import packages
library(tidyverse) #for data wrangling
library(here) #for filepath management
library(ggsci) #for colors
library(scales) #for log axis breaks

#custom graphing theme
#including geoms to reduce repetition
theme <- list(
  theme_classic(),
  scale_color_jama(),
```

```
  scale_fill_jama(),
  theme(legend.position="right"),
  labs(x="Year",y="Population")
)
```

2 Part I

2.1 Task 1

Write a custom function to calculate the value of $\frac{dN}{dt} = rN(t)(1 - N(t)/k) - EN(t)$. The function should take as arguments the current stock-size N, level of fishing effort E and model parameters r, k.

```
#function with some default parameters
dNdt_BM_logistic_CE <- function(N=0.25, r=2.5, K=1, E=0.5){

  #calculate dN_dt
  r * N * (1-(N/K))-(E*N)

}
```

Use the custom R function to plot the growth rate dN/dt for a few different sets of values for parameters r and K. How do the parameters affect the shape?

```
#empty list to store results
result.list <- list()

#nested for loops
#range of r values
for (r in c(1,2,3)){

  #range of K values
  for (K in c(1,2)){

    for (N in seq(0,2,0.01)){


      #calculate dN/dt
      dN_dt <- dNdt_BM_logistic_CE(N=N, r=r, K=K, E=0)

      #dataframe to store results
      df <- data.frame(dN_dt=dN_dt, N=N, r=as.character(r), K=as.character(K))

      #store results
      result.list <- append(result.list, list(df))

    } #N
  } #k
} #r

#compile results
```

```
result.df <- bind_rows(result.list) %>%
  filter(dN_dt>=0)

#plot
ggplot(result.df, aes(N, dN_dt, color=r, linetype=K))+
  theme+
  geom_line()+
  scale_y_continuous(expand=expansion(mult=c(0,0.05)))+
  labs(x="N", y="dN/dt")
```

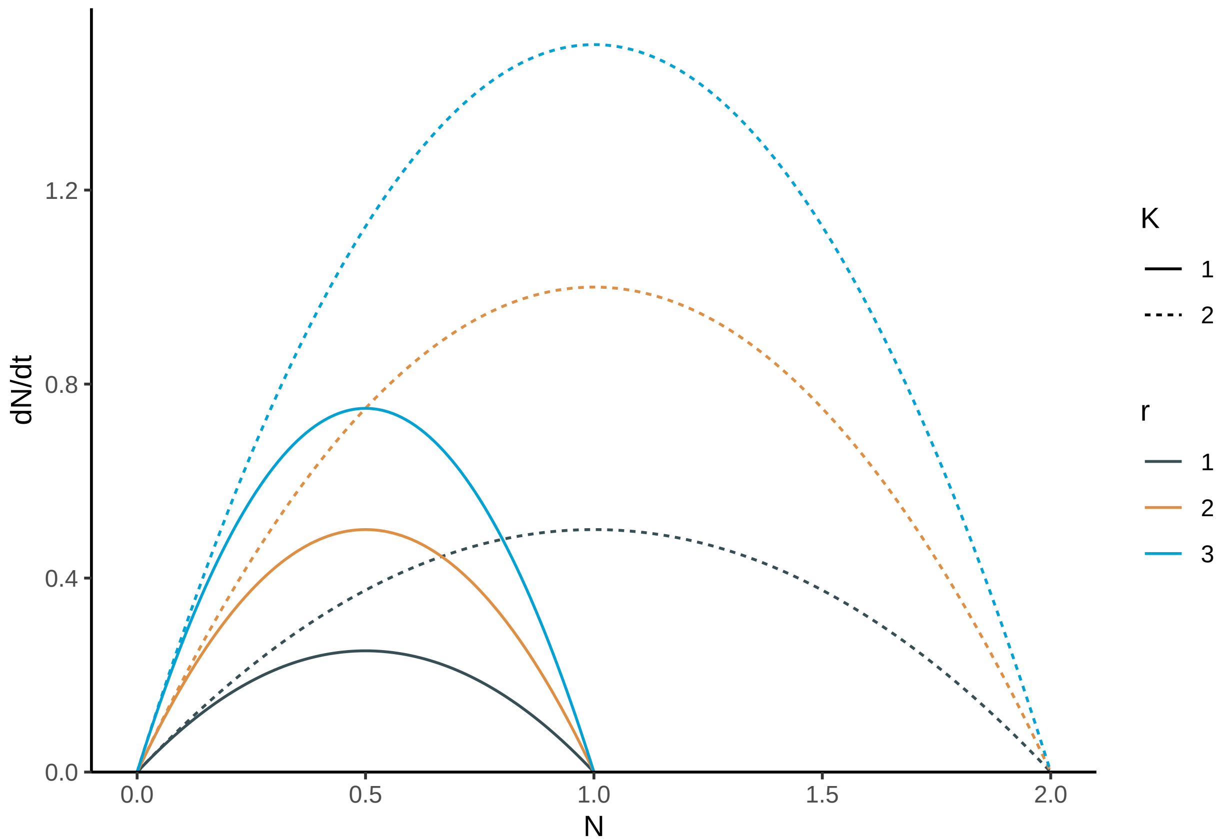


Fig. 1: dN/dt increases with both r and K, and the maximum dN/dt occurs when $N = 1/2K$

2.2 Task 2

Rewrite the `for` loop above to simulate the population dynamics but use the custom function to get the rate of change of the population.

```
#empty list to store results
result.list <- list()
#duration of iteration
dt <- 0.01

#nested for loops
#range of r values
```

```
for (r in c(1,2,3)){

  #range of K values
  for (K in c(1,2)){

    #reset N0
    N <- 0.25

    #iterate over 10 years (if dt is in units of years)
    for (t in c(1:(10/dt))){

      #calculate N
      N <- N + dt*dNdt_BM_logistic_CE(N=N, r=r, K=K, E=0)

      #dataframe to store results
      df <- data.frame(t=t*dt, N=N, r=as.character(r), K=as.character(K))

      #store results
      result.list <- append(result.list, list(df))

    } #time
  } #k
} #r

#compile results
result.df <- bind_rows(result.list)

#plot
ggplot(result.df, aes(t, N, color=r, linetype=K))+
  theme+
  geom_line()
```

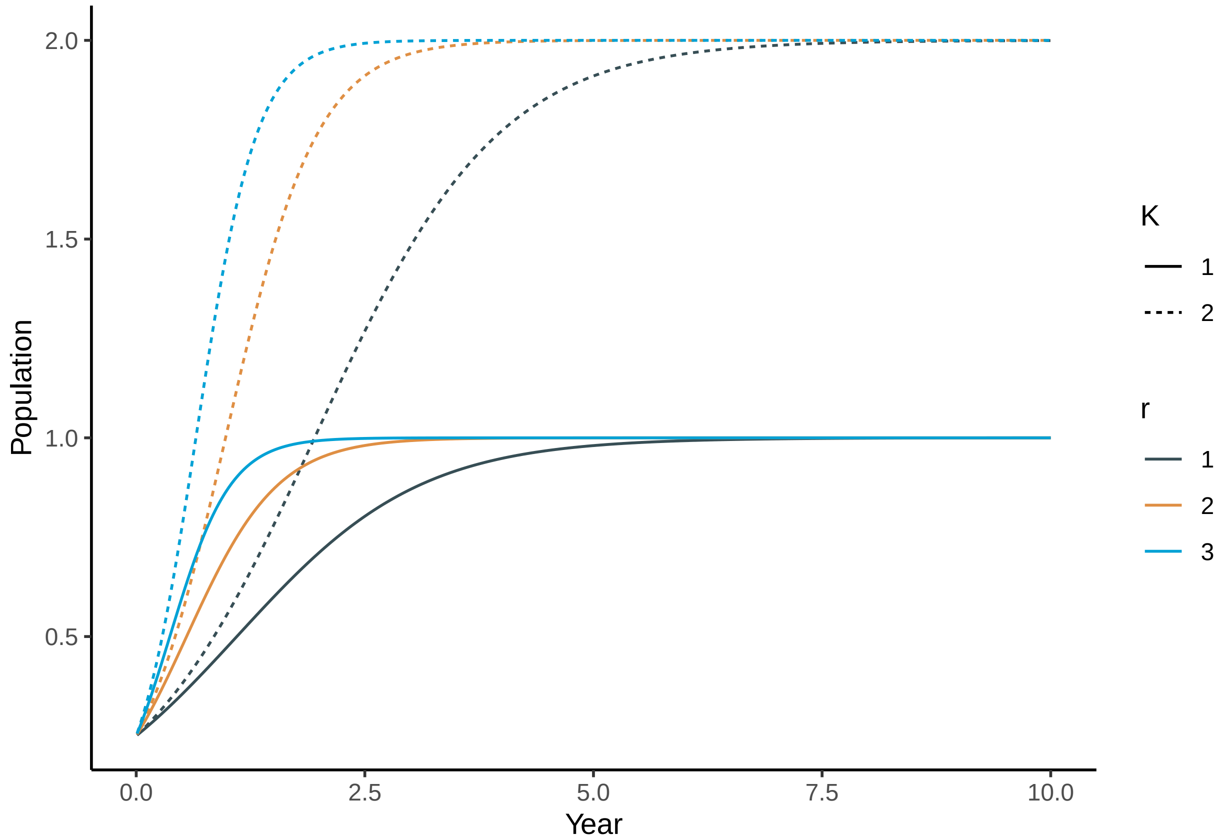


Fig. 2: Population asymptotically approaches K, and a larger K results in a larger population at equilibrium r influences the "steepness" of the population curve: at higher replacement, the return from the purtubation is faster.

2.3 Task 3

```
#make a function with some default parameters
f_sim_BM_logistic_CE <- function(N0=0.25, r=2.5, K=1, E=0.5, dt=0.01, duration=10){

  #empty list to store results
  result.v <- vector()

  #set initial population
  N <- N0

  #iterate
  for (t in c(1:(duration/dt))){

      #calculate N
      N <- N + dt*dNdt_BM_logistic_CE(N=N, r=r, K=K, E=E)

      #can't have a negative population!
      if (N<0){
        N <- 0
```

```r
        }

      #dataframe to store results
      result.v <- c(result.v, N)

  }

  return(result.v)
}

result.list <- list()

for (E in c(0.2, 0.5, 0.7)){

  N.v <- f_sim_BM_logistic_CE(E=E)

  df <- data.frame(N = N.v,
                   E=as.character(E)) %>%
    mutate(t=row_number()*dt)

  result.list <- append(result.list, list(df))

}

result.df <- bind_rows(result.list)

#plot
ggplot(result.df, aes(t, N, color=E))+
  theme+
  geom_line()
```

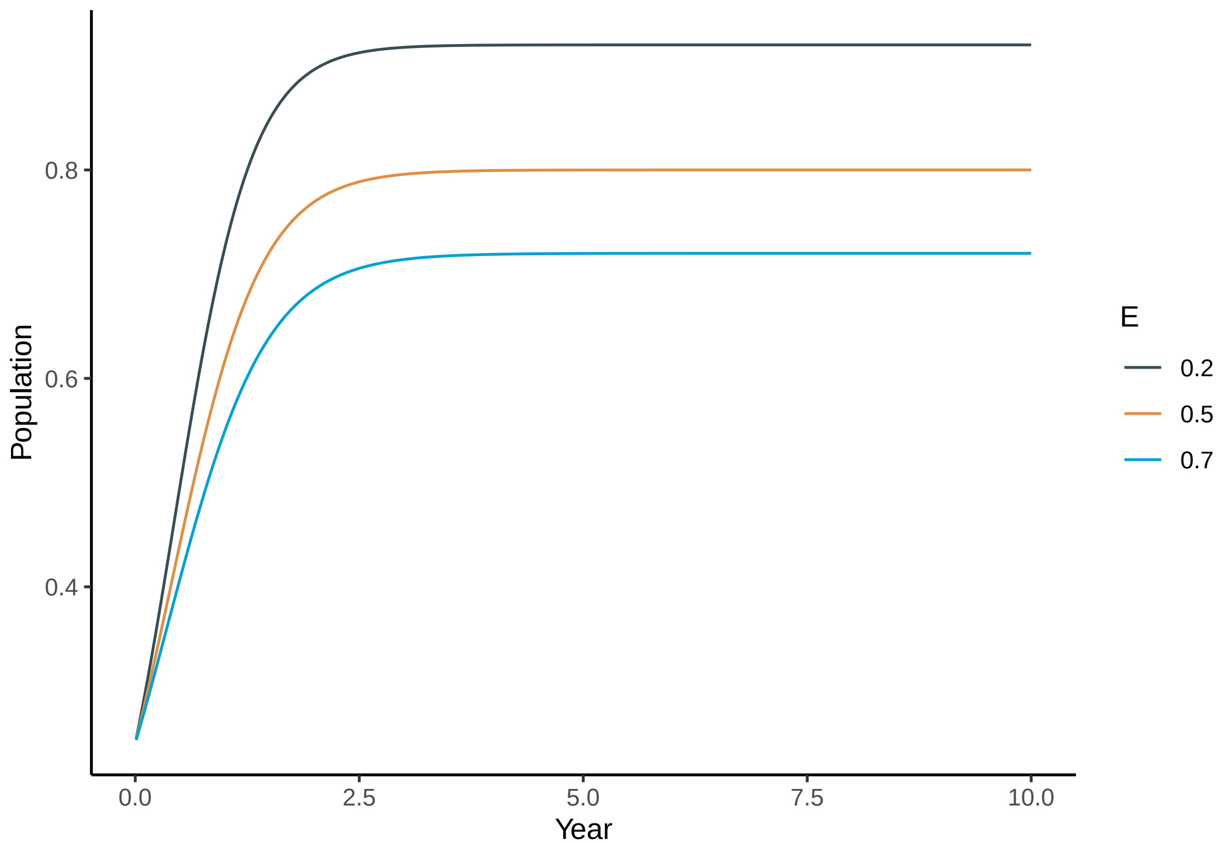


Fig. 3: E impacts the equilibrium population size, with larger E corresponding to a lower population size and slower recovery.

2.4 Task 4

```
#function to fine equilibroum population level
f_N_equil_E <- function(K=1,E=0.5,r=2.5){
  f_N_equil_E <- K * (1 - (E/r))

  #can't have a negative population!
  if (f_N_equil_E<0){
    f_N_equil_E <- 0
  }

  return(f_N_equil_E)
}

#function for single perturbation
return_time_CE <- function(dN=0.2, r=2.5, K=1, E=0.2){

  N_f_N_equil_E <- f_N_equil_E(K,E,r)
  N0 <- N_f_N_equil_E - dN
  v_N <- f_sim_BM_logistic_CE(N0=N0, r=r, K=K, E=E)
```

```
  #deal with the infinity issue
  #if the population has crashed, there is no return time
  if (v_N[1]>v_N[length(v_N)] | v_N[length(v_N)]==0){
    return_index <- NA
  } else {
    return_index <- min(which(abs(v_N - N_f_N_equil_E) < dN / exp(1)))
  }

  return(return_index)
}

#plot of population over time
plot.df <- data.frame(N=f_sim_BM_logistic_CE(E=0.2)) %>%
  mutate(t=row_number()*dt)

#find return time
point <- c(return_time_CE(E=0.2)*dt,plot.df[return_time_CE(E=0.2),"N"])

#make the plot
ggplot(plot.df, aes(t, N))+
  theme+
  geom_line()+
  geom_point(x=point[1], y=point[2], shape=23, size=4, fill="darkred")+
  geom_hline(yintercept=f_N_equil_E(1,0.2,2.5), linetype="dashed")
```

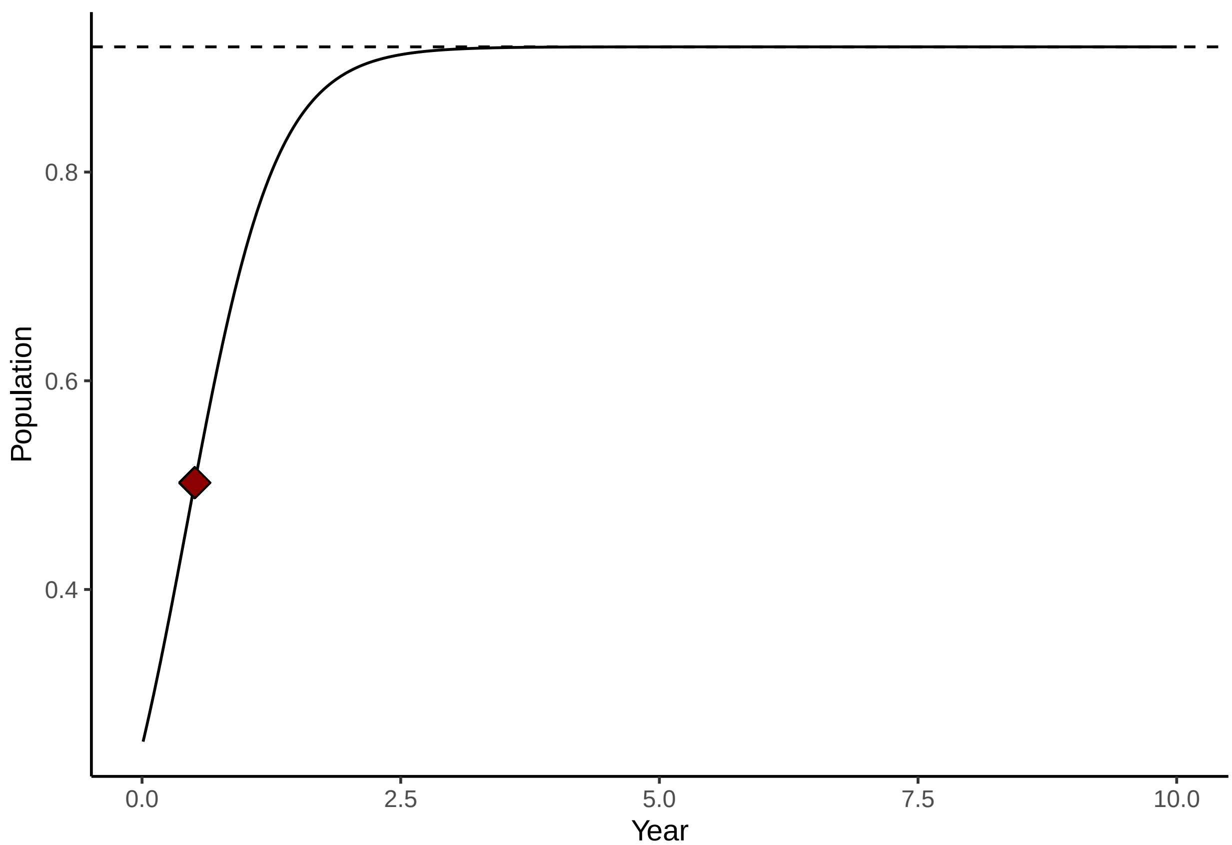


Fig. 4: Plot of return time with $E = 0.2$.

2.5 Task 5

```
#empty vectors to store results
rt.v <- vector()
yield.v <- vector()

#for loop
for (E in seq(0,2,0.1)){

  N_f_N_equil_E <- f_N_equil_E(E=E)
  rt <- return_time_CE(dN=0.1, E=E)*dt

  yield.v <- c(yield.v,N_f_N_equil_E*E)
  rt.v <- c(rt.v,rt)
}

result.df <- data.frame(rt=rt.v/rt.v[1], yield=yield.v)

ggplot(result.df, aes(yield,rt))+
  theme+
  geom_line(orientation = "y")+
  labs(x="Yield", y="Return Time")
```

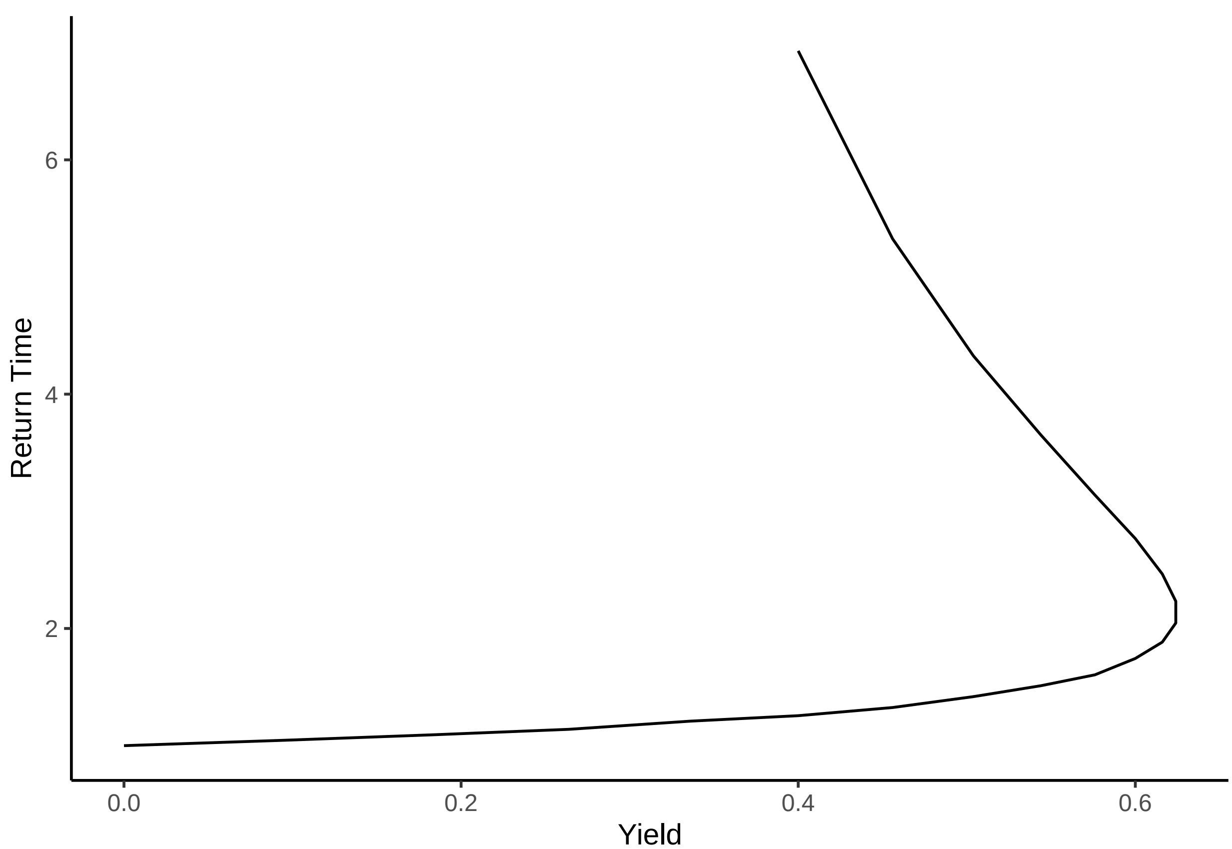


Fig. 5: Change in return time with yield.

2.6 Task 6

- `dNdt_BM_logistic_CE()`: This function, which calculates dN/dt, needs to be modified
- For loops in tasks 1 and 2 need to be modified to use a different function from `dNdt_BM_logistic_CE()`
- `f_sim_BM_logistic_CE()`: This function, which returns a vector of N over time, needs to be modified to not use `dNdt_BM_logistic_CE()`
- `f_N_equil_E()`: This function, which calculates the equilibrium population level, needs to be modified
- `return_time_CE()`: this function, which calculates the return time of the population from a perturbation, needs to be modified to not use `f_N_equil_E()` and `f_sim_BM_logistic_CE()`

3 Part II

3.1 Task 7

```
#repeat everything for constant yield

#function for dN/dt
dNdt_BM_logistic_CY <- function(N=0.25, r=2.5, K=1, Y=0.1){

  #calculate dN_dt
  r * N * (1-(N/K))-Y

}

#logistic growth model
f_sim_BM_logistic_CY <- function(N0=0.25, r=2.5, K=1, Y=0.1, dt=0.01, duration=10){

  #empty list to store results
  result.v <- vector()

  #set initial population
  N <- N0

  #iterate
  for (t in c(1:(duration/dt))){

      #calculate N
      N <- N + dt*dNdt_BM_logistic_CY(N=N, r=r, K=K, Y=Y)

      #can't have a negative population!
      if (N<0){N <- 0}

      #dataframe to store results
      result.v <- c(result.v, N)

  }

  return(result.v)
```

```
}

#function to find equilibrium population level
f_N_equil_Y <- function(K=1,Y=0.1,r=2.5){

  #do this part separately to identify population crash
  intermediate <- 1-(4*Y)/(r*K)

  #fix the infinity issue
  if (intermediate<0){
    intermediate <- 0
  }

  N_equil <- K/2 * (1 + sqrt(intermediate))

  return(N_equil)
}

#function for single perturbation
return_time_CY <- function(dN=0.2, r=2.5, K=1, Y=1){

  N_f_N_equil_Y <- f_N_equil_Y(K=K,Y=Y,r=r)
  N0 <- N_f_N_equil_Y - dN
  v_N <- f_sim_BM_logistic_CY(N0=N0, r=r, K=K, Y=Y)

  #deal with the infinity issue
  #if the population does not approach equilibrium by the end of the simulation, return NA
  if (v_N[1]>v_N[length(v_N)] | v_N[length(v_N)]==0){
    return_index <- NA
  } else {
    return_index <- min(which(abs(v_N - N_f_N_equil_Y) < dN / exp(1)))
  }

  return(return_index)
}

#make a quick plot to see how population dynamics change with yield
result.list <- list()

#iterate over three values of y
for (Y in c(0.3, 0.4, 0.5)){

  N.v <- f_sim_BM_logistic_CY(N0=0.3, Y=Y)

  df <- data.frame(N = N.v,
                   Y=as.character(Y)) %>%
    mutate(t=row_number()*dt)

  result.list <- append(result.list, list(df))

}

result.df <- bind_rows(result.list)
```

```
#plot
ggplot(result.df, aes(t, N, color=Y))+
  theme+
  geom_line()
```

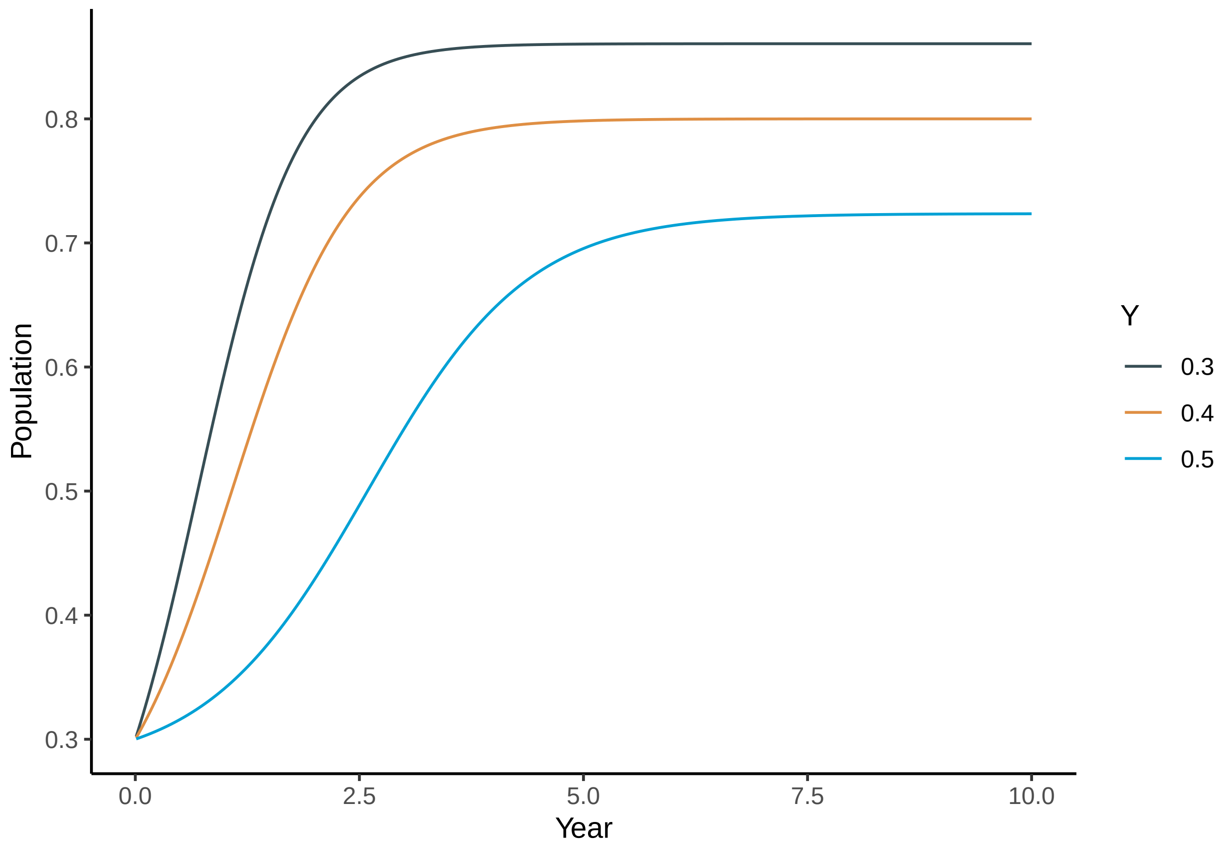


Fig. 6: Population dynamics under different yields. The population equilibrium decreases with increasing yield.

3.2 Task 8

```
#empty vectors to store results
rt.v <- vector()
yield.v <- vector()

#for loop
for (Y in seq(0,1,0.01)){

  N_f_N_equil_Y <- f_N_equil_Y(Y=Y)
  rt <- return_time_CY(dN=0.1, Y=Y)

  yield.v <- c(yield.v,Y)
  rt.v <- c(rt.v,rt)
```

```
}

result.df <- data.frame(rt=rt.v/rt.v[1], yield=yield.v) %>%
  drop_na()

ggplot(result.df, aes(yield,rt))+
  theme+
  geom_line(orientation = "y")+
  labs(x="Yield", y="Return Time")
```

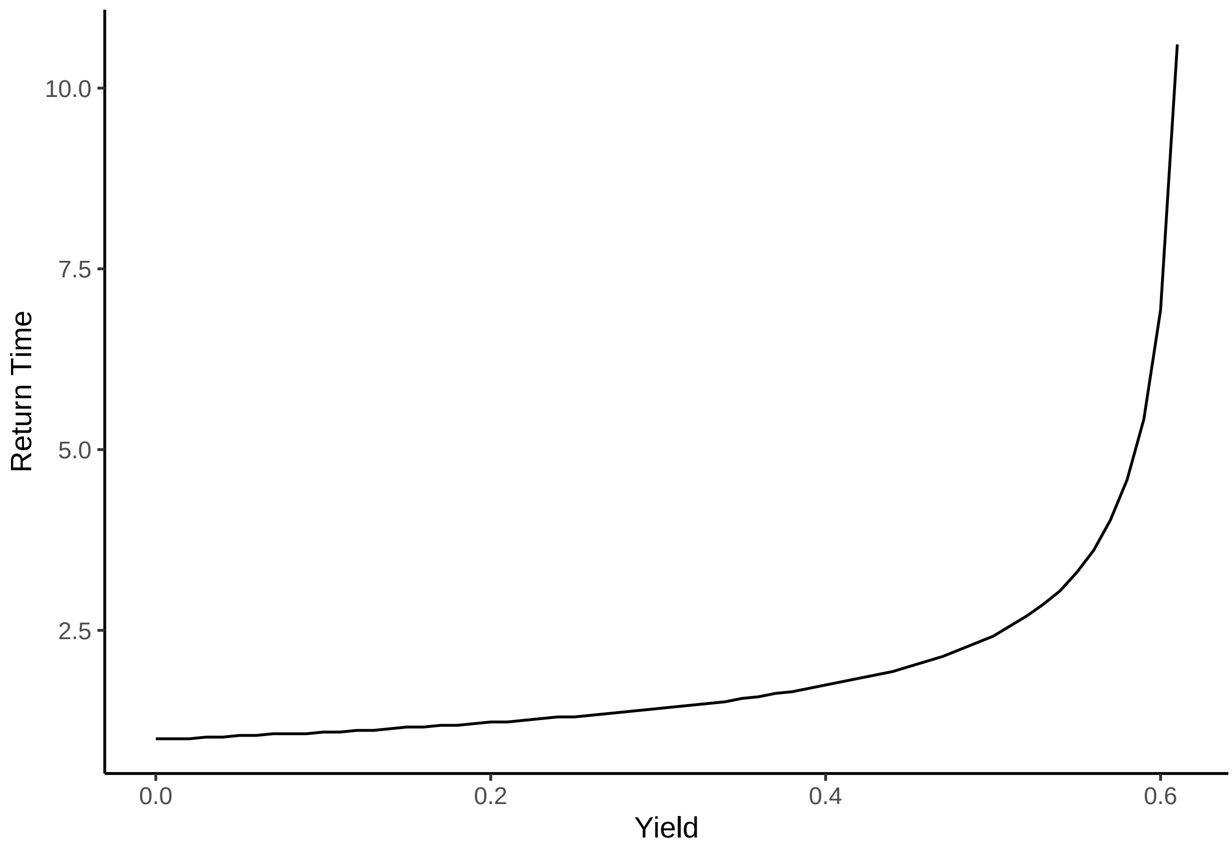


Fig. 7: Variations in return time with yield. Increasing yield results in an increasing return time.

3.3 Task 9

```
#wrap this in a function because we're going to do it again in task 10
model_return_time <- function(K=1,r=2.5){

  #empty list
  result.list <- list()

  for (type in c("Effort","Yield")){
```

```
  if (type == "Effort"){

      for (E in seq(0,2,0.01)){

      N_f_N_equil_E <- f_N_equil_E(K=K, r=r, E=E)
      rt <- return_time_CE(dN=0.1, K=K, r=r, E=E)*dt

      df <- data.frame(yield = N_f_N_equil_E*E,
                       return_time = rt,
                       type = type)

      result.list <- append(result.list, list(df))

    } #for Y
  } #if effort

  if (type == "Yield"){

    for (Y in seq(0,1,0.01)){

      N_f_N_equil_Y <- f_N_equil_Y(K=K, r=r, Y=Y)
      rt <- return_time_CY(dN=0.1, K=K, r=r, Y=Y)

      df <- data.frame(yield = Y,
                       return_time = rt,
                       type = type)

      result.list <- append(result.list, list(df))

    } #for Y
  } #if yield
} #for type

#compile dataframe
result.df <- bind_rows(result.list)

#normalize effort and yield return times
effort.v <- result.df %>%
  filter(type=="Effort") %>%
  pull(return_time)

effort.v <- effort.v/effort.v[1]

yield.v <- result.df %>%
  filter(type=="Yield") %>%
  pull(return_time)

yield.v <- yield.v/yield.v[1]

#add normalized return times
plot.df <- result.df %>%
  mutate(norm_rt = c(effort.v,yield.v))
```

```
  return(plot.df)

}

plot.df <- model_return_time()

#plot return time
ggplot(plot.df, aes(yield, norm_rt, color=type))+
  theme+
  geom_line(orientation = "y")+
  labs(x="Yield", y="Return Time", color="Model Type")
```

```
## Warning: Removed 39 rows containing missing values or values outside the scale range
## (‘geom_line()‘).
```

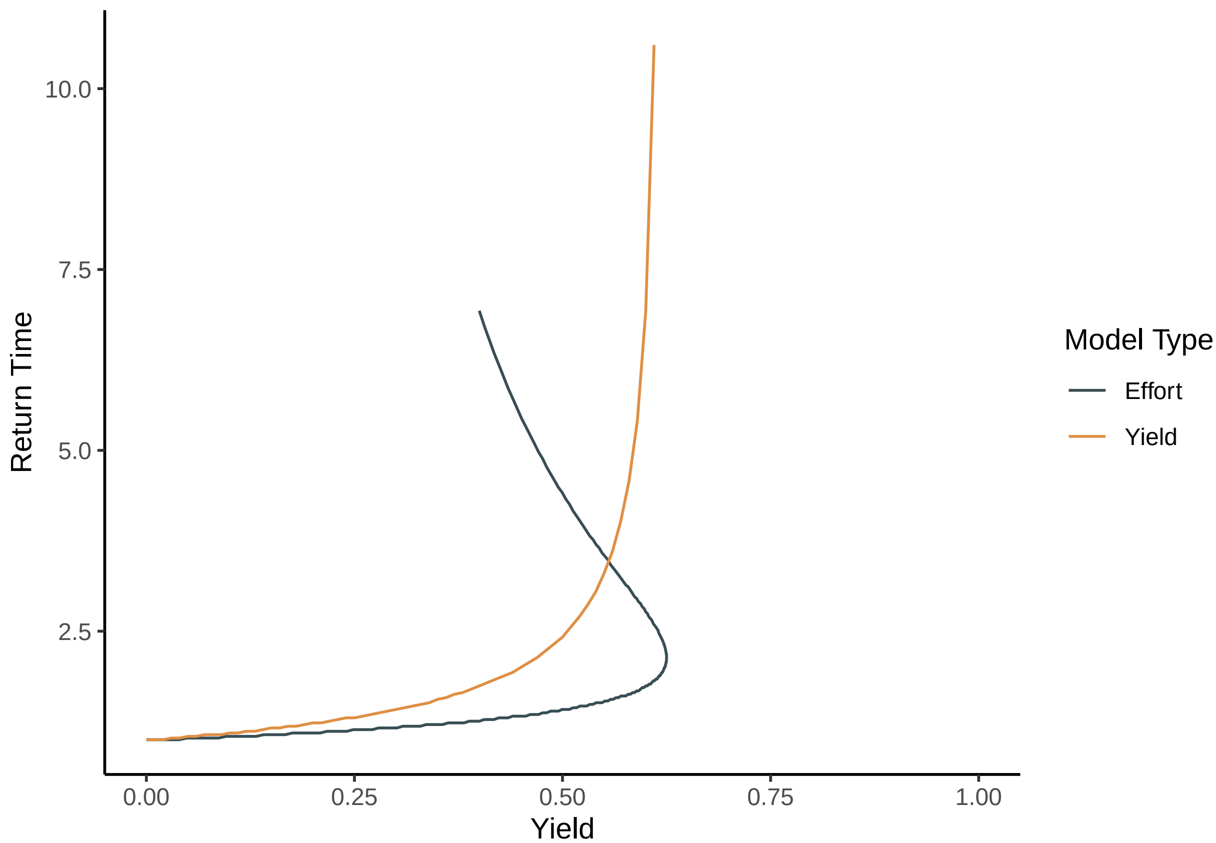


Fig. 8: Replication of Fig. 2.

3.4 Task 10

```
result.list <- list()

for (r in c(0.5,1,1.5)){
```

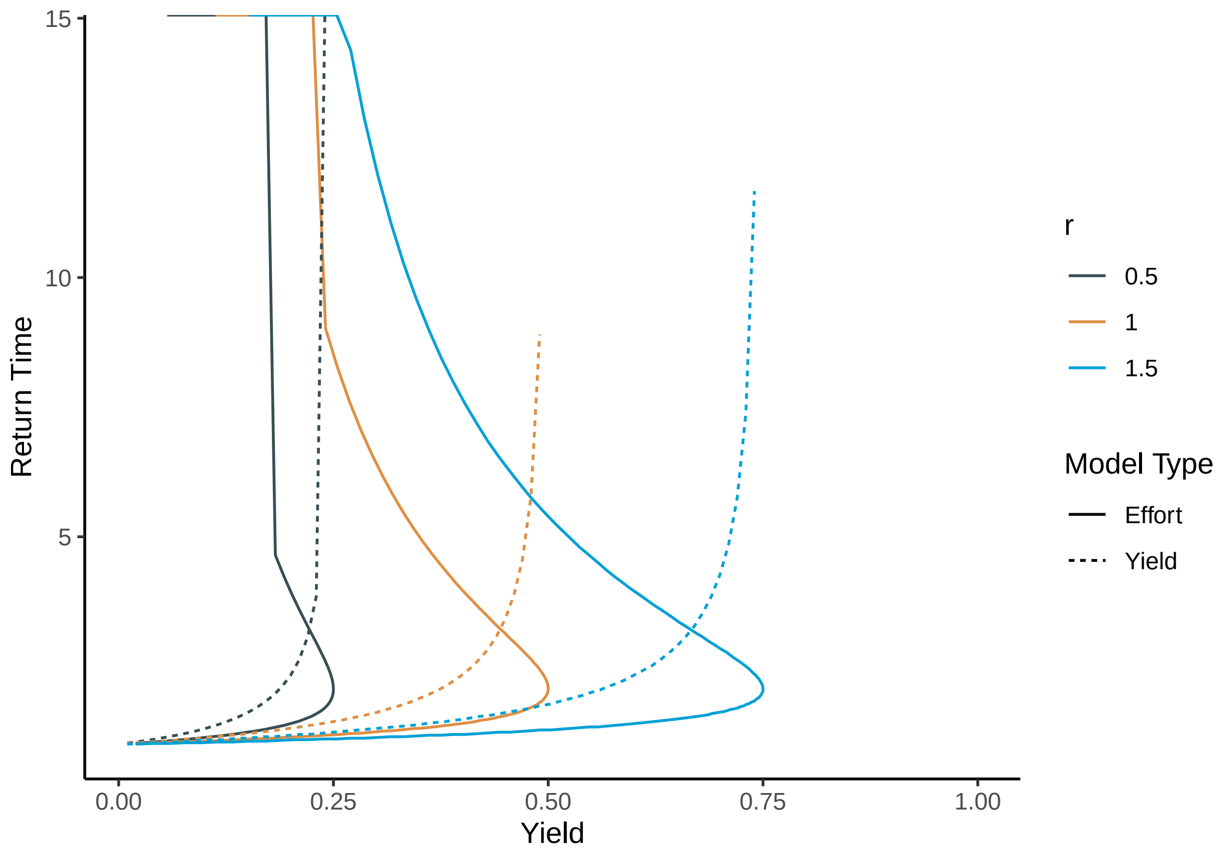
```r
  df <- model_return_time(K=K,r=r) %>%
    mutate(K=as.character(K),
           r=as.character(r))

  result.list <- append(result.list, list(df))

} #for r
```

```
## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_Y) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf
```

```
## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf

## Warning in min(which(abs(v_N - N_f_N_equil_E) < dN/exp(1))): no non-missing
## arguments to min; returning Inf
```

```
result.df <- bind_rows(result.list) %>%
  filter(is.numeric(norm_rt),
         yield>0)

#plot return time
ggplot(result.df, aes(yield, norm_rt, linetype=type, color=r))+
  theme+
  geom_line(orientation = "y")+
  labs(x="Yield", y="Return Time", linetype="Model Type")
```

```
## Warning: Removed 167 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



15
10
5
Return Time
0.00
0.25
0.50
0.75
1.00
Yield
r
0.5
1
1.5
Model Type
Effort
Yield