Problem 3   Lisa Wang
UID: 105502901

Question 1:

ⓐ part A.

r_d = per capita rate of growth b_o = mean birth rate of population

$R = 1 + r_d = 1.4$ year , in discrete time

$R = (1-d)(1+b) = (1-d_o)(1+0.6) = 1.4$

$d_o = 0.125$

ⓑ $b(n) = b_o - b_1 n$, birth rate b depends linearly on population size

since $R = 1 + r_d$, we know that $R = (1-d)(1+b)$

$r_d = (1-d)(1+b) - 1$ $r_d = (1-d_o)(1+(b_o - b_1 n)) - 1$

$$\boxed{r_d = b_o - d_o - d_o b_o}$$

$r_d = (1-d_o)(1+(b_o - b_1 K)) - 1$ where r_d is 0 when carrying capacity

$0 = (1-d_o)(1 + b_o - b_1 K) - 1$

$$\boxed{K = \frac{b_o - b_o d_o - d_o}{b_1(1-d_o)}}$$

The system conforms to the definition of the logistical growth model (discrete) because in class, we discussed that the logistic discrete-time model is under the assumption that crowding occurs and resources are limiting. Here, insect population is dependent on rainfall, which means the resource is limited and that the birth rate is density-dependent is also a good indicator.

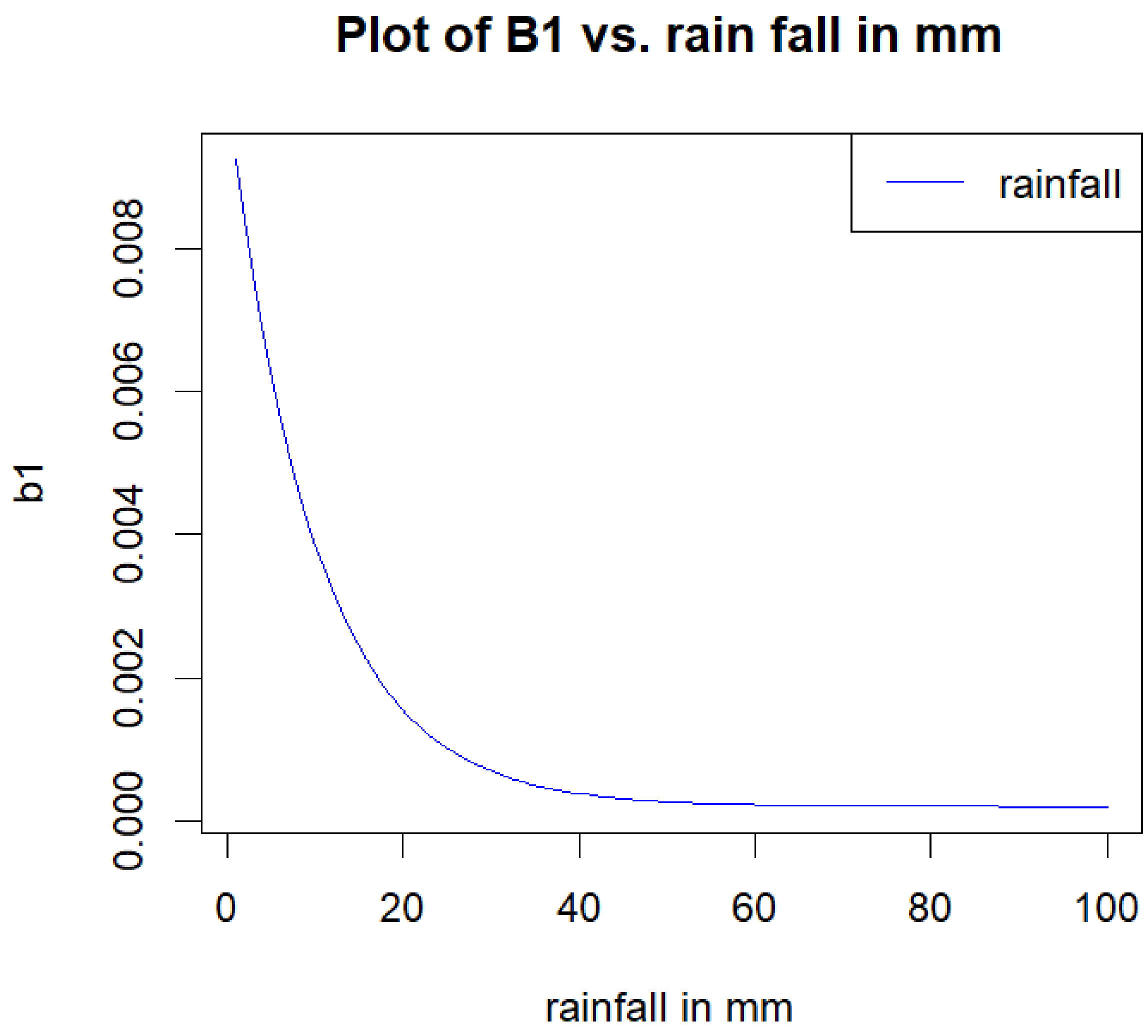
Given log-discrete-time model, if per capita rate r_d is positive but not too high, then pop → K (reaches capacity). At low density, population drops to 0. If per capita ~~of~~ rate r_d's large enough and positive, population will oscilliate indefinitely.

ⓒ The b_1 parameter is the slope of the birth rate vs. population graph, where a more densely populated population will be more impacted by the lack of resource due to overt competition, and thereby have a smaller birth rate b(n). b_1 is the rate at which the increase of one individual in the population have on the birth rate of that density-dependent system.

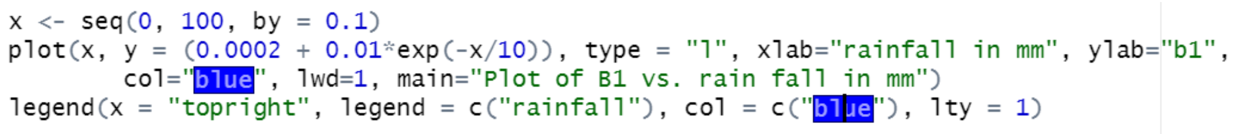
$B(n) = b_o - b_1 n$

slope of graph $= m = -b$

Problem Set 3
Question 1d.



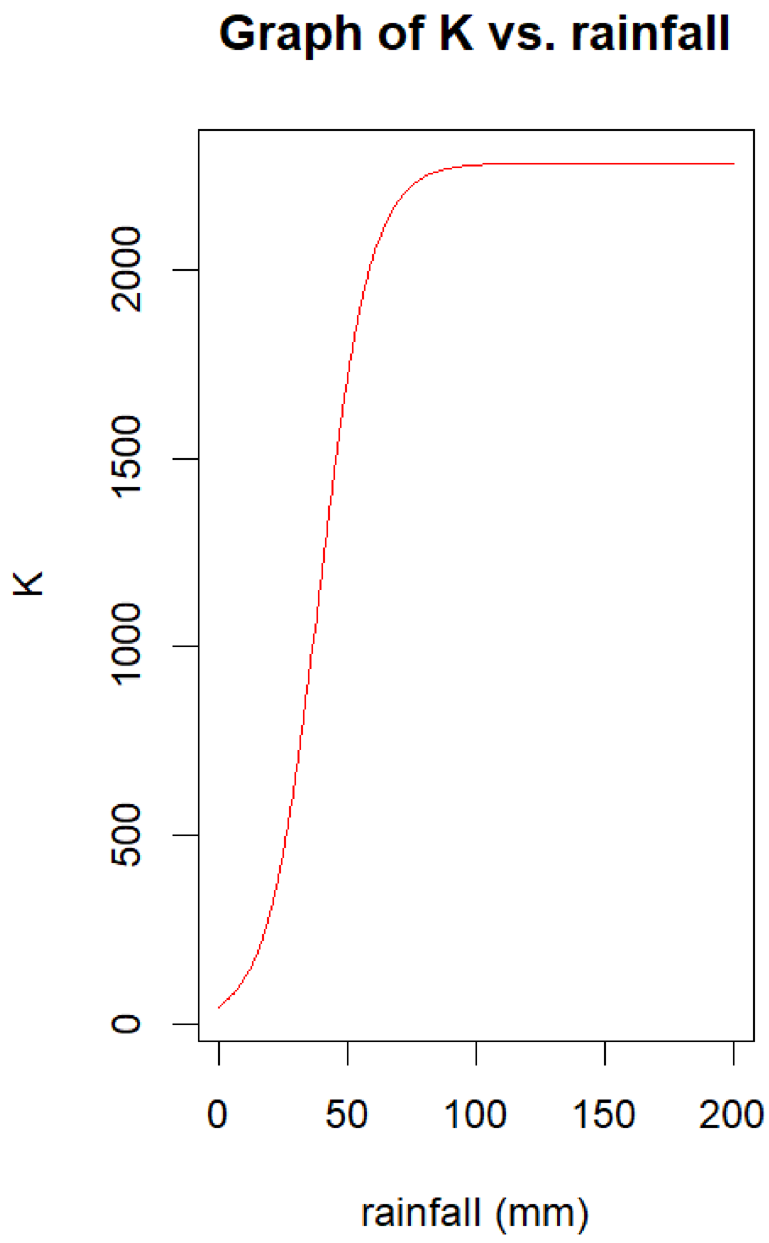
Code:



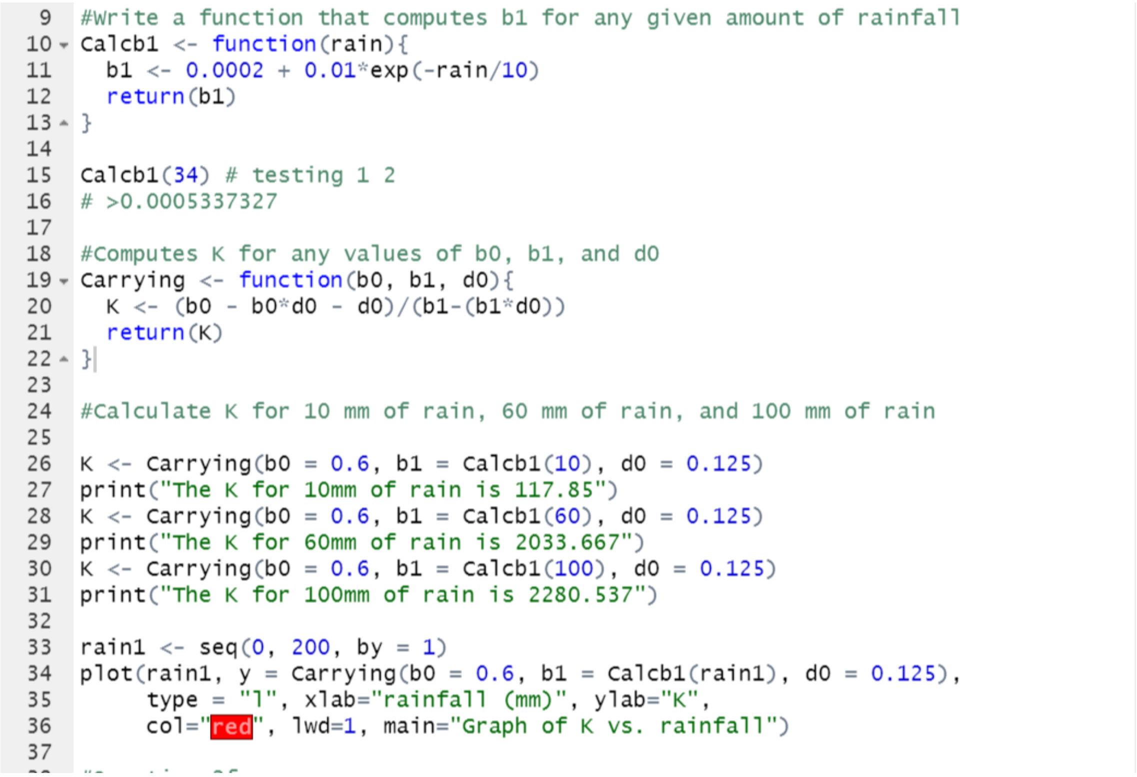
```
x <- seq(0, 100, by = 0.1)
plot(x, y = (0.0002 + 0.01*exp(-x/10)), type = "l", xlab="rainfall in mm", ylab="b1",
        col="blue", lwd=1, main="Plot of B1 vs. rain fall in mm")
legend(x = "topright", legend = c("rainfall"), col = c("blue"), lty = 1)
```

```
#1d------------------
x <- seq(0, 100, by = 0.1)
plot(x, y = (0.0002 + 0.01*exp(-x/10)), type = "l", xlab="rainfall in mm", ylab="b1",
     col="blue", lwd=1, main="Plot of B1 vs. rain fall in mm")
legend(x = "topright", legend = c("rainfall"), col = c("blue"), lty = 1)
```

1e.


Graph of K vs. rainfall
K
0
500
1000
1500
2000
0
50
100
150
200
rainfall (mm)

Code:

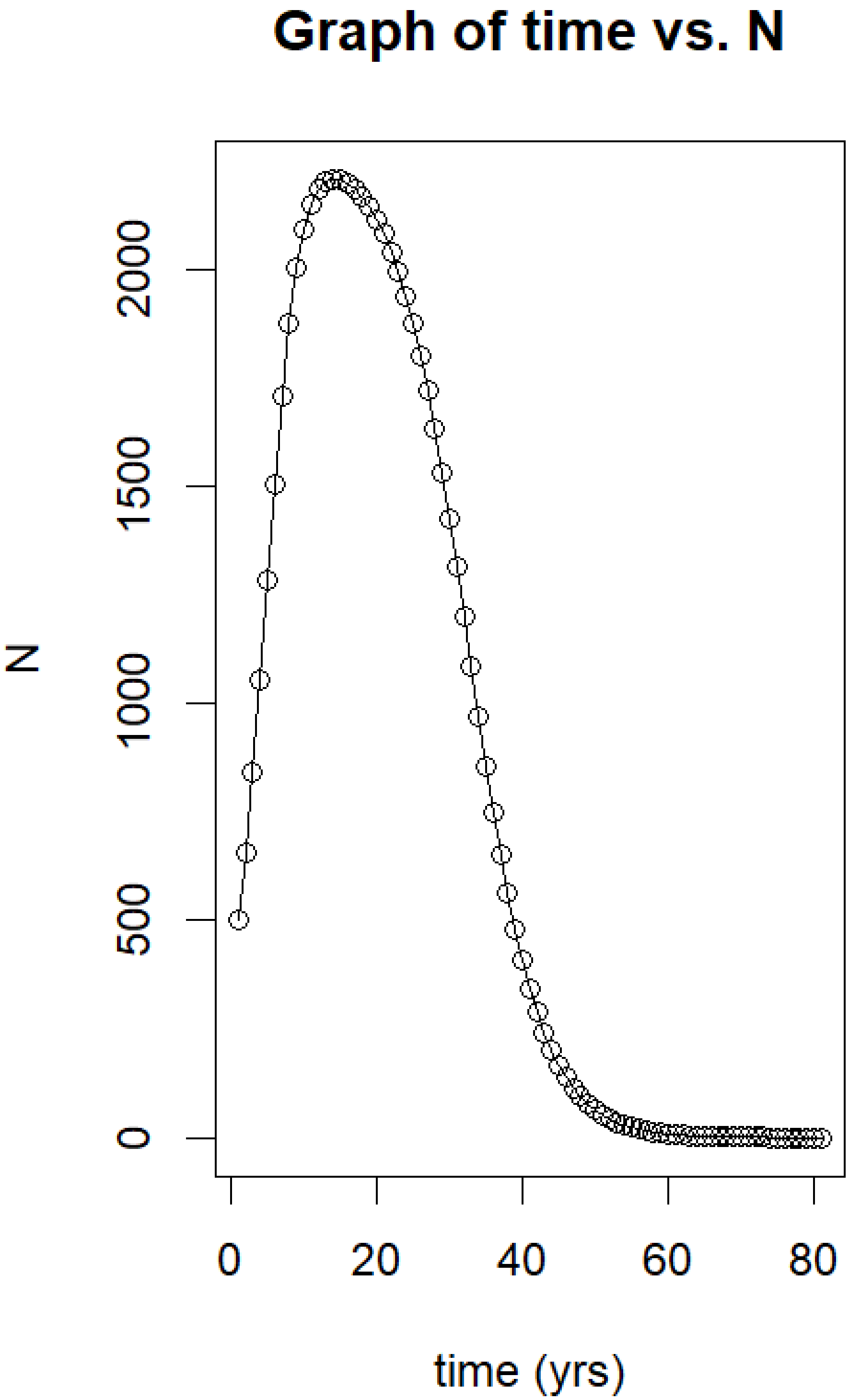


```r
#Write a function that computes b1 for any given amount of rainfall
Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Calcb1(34) # testing 1 2
# >0.0005337327

#Computes K for any values of b0, b1, and d0
Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

#Calculate K for 10 mm of rain, 60 mm of rain, and 100 mm of rain

K <- Carrying(b0 = 0.6, b1 = Calcb1(10), d0 = 0.125)
print("The K for 10mm of rain is 117.85")
K <- Carrying(b0 = 0.6, b1 = Calcb1(60), d0 = 0.125)
print("The K for 60mm of rain is 2033.667")
K <- Carrying(b0 = 0.6, b1 = Calcb1(100), d0 = 0.125)
print("The K for 100mm of rain is 2280.537")

rain1 <- seq(0, 200, by = 1)
plot(rain1, y = Carrying(b0 = 0.6, b1 = Calcb1(rain1), d0 = 0.125),
     type = "l", xlab="rainfall (mm)", ylab="K",
     col="red", lwd=1, main="Graph of K vs. rainfall")

```

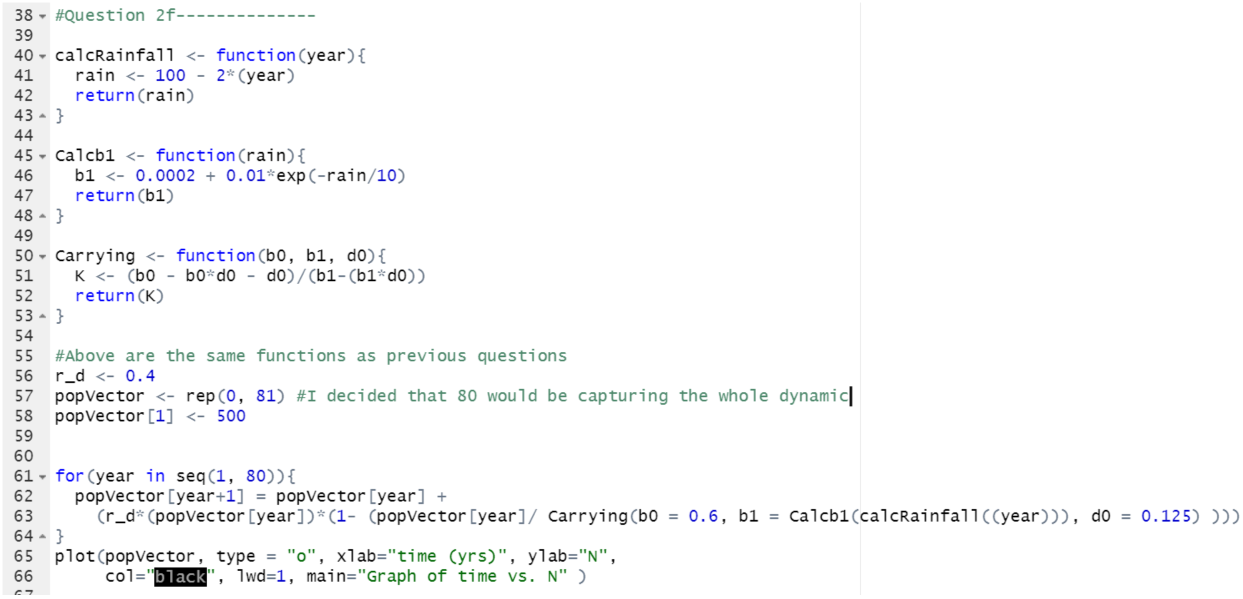
```r
#1e-------------------------
#Write a function that computes b1 for any given amount of rainfall
Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Calcb1(34) # testing 1 2
# >0.0005337327

#Computes K for any values of b0, b1, and d0
Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

#Calculate K for 10 mm of rain, 60 mm of rain, and 100 mm of rain

K <- Carrying(b0 = 0.6, b1 = Calcb1(10), d0 = 0.125)
print("The K for 10mm of rain is 117.85")
K <- Carrying(b0 = 0.6, b1 = Calcb1(60), d0 = 0.125)
print("The K for 60mm of rain is 2033.667")
```

```
K <- Carrying(b0 = 0.6, b1 = Calcb1(100), d0 = 0.125)
print("The K for 100mm of rain is 2280.537")

rain1 <- seq(0, 200, by = 1)
plot(rain1, y = Carrying(b0 = 0.6, b1 = Calcb1(rain1), d0 = 0.125),
     type = "l", xlab="rainfall (mm)", ylab="K",
     col="red", lwd=1, main="Graph of K vs. rainfall")
```

1f.



Code:

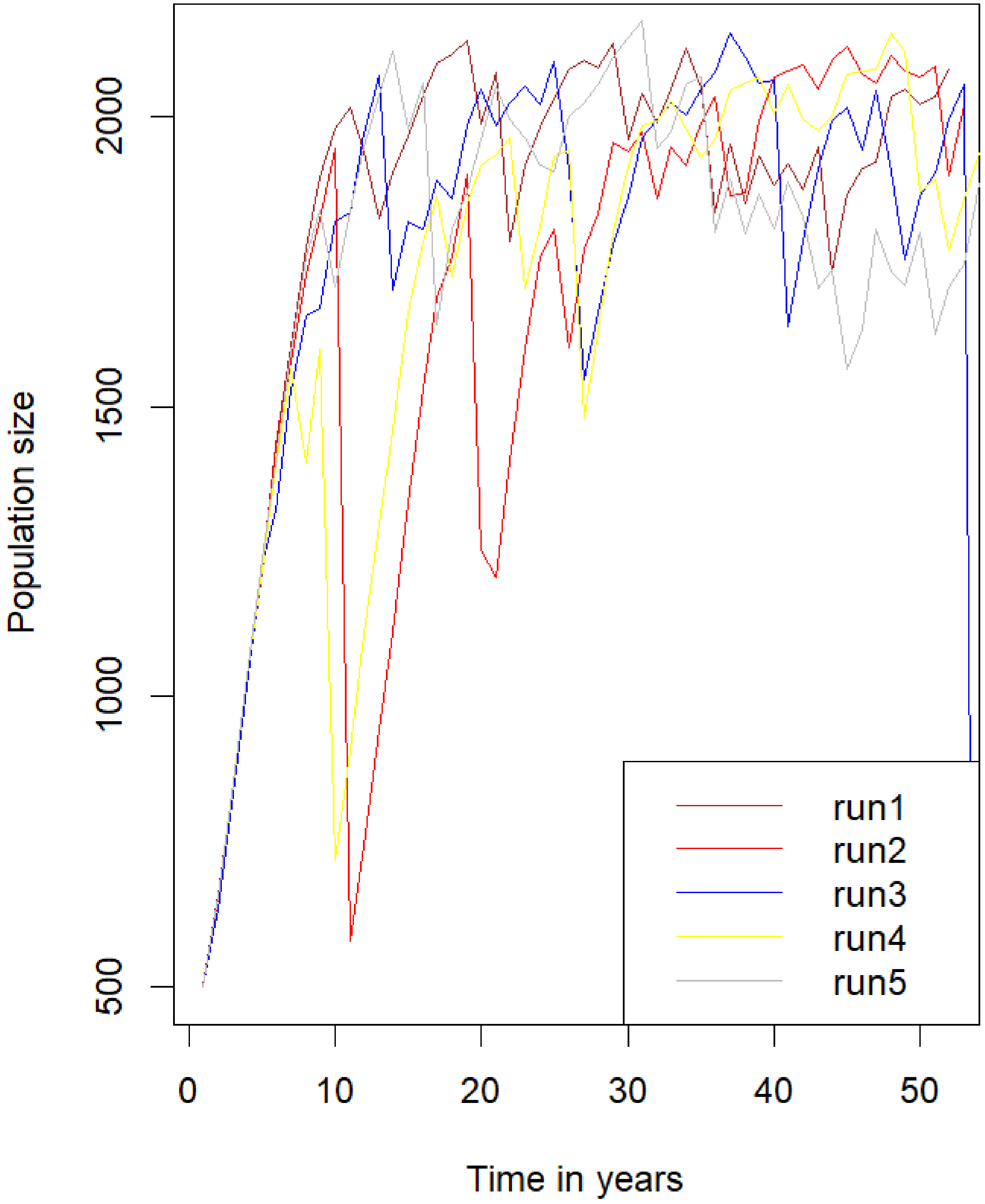


```
#Question 2f--------------

calcRainfall <- function(year){
  rain <- 100 - 2*(year)
  return(rain)
}

Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

#Above are the same functions as previous questions
r_d <- 0.4
popVector <- rep(0, 81) #I decided that 80 would be capturing the whole dynamic
popVector[1] <- 500


for(year in seq(1, 80)){
  popVector[year+1] = popVector[year] +
    (r_d*(popVector[year])*(1- (popVector[year]/ Carrying(b0 = 0.6, b1 = Calcb1(calcRainfall((year))), d0 = 0.125) )))
}
plot(popVector, type = "o", xlab="time (yrs)", ylab="N",
     col="black", lwd=1, main="Graph of time vs. N" )
```

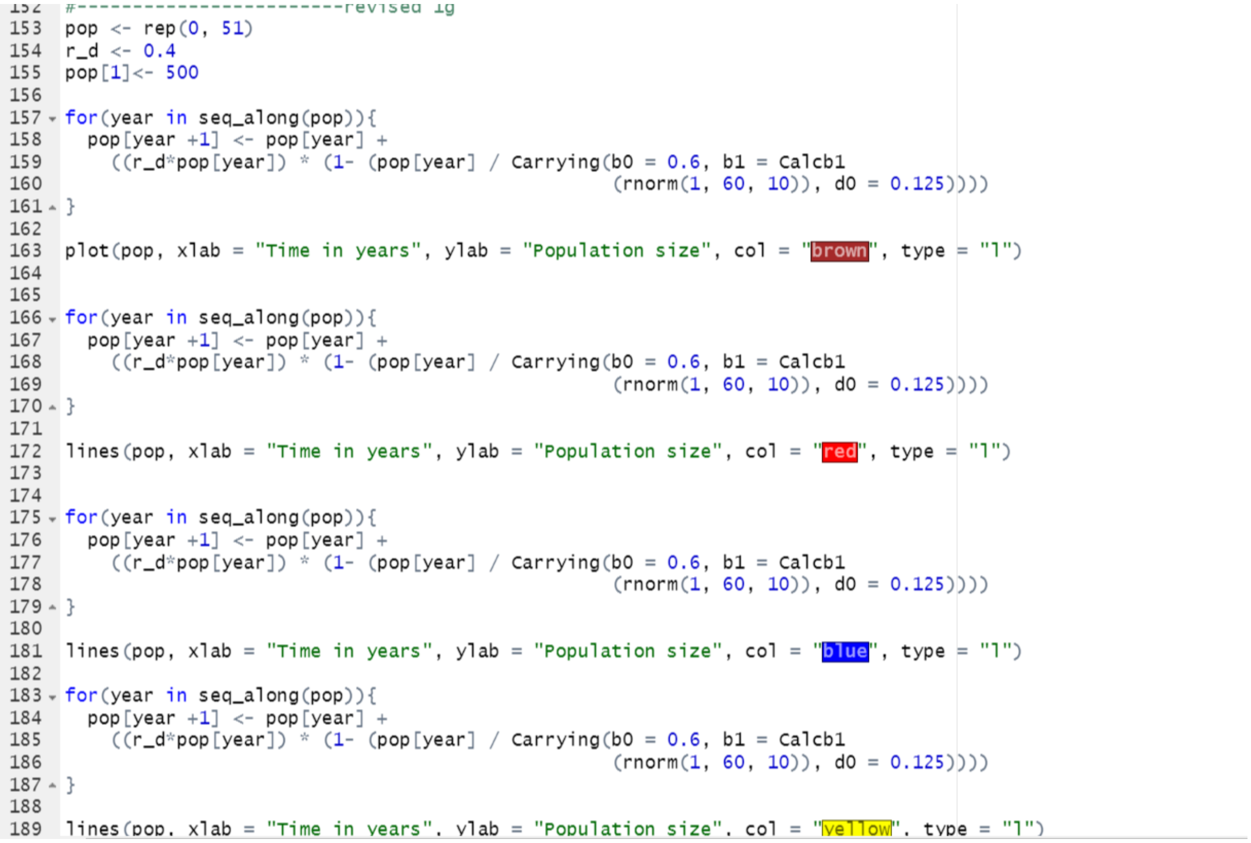
```
#Question 1f--------------

calcRainfall <- function(year){
  rain <- 100 - 2*(year)
  return(rain)
}

Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

#Above are the same functions as previous questions
r_d <- 0.4
popVector <- rep(0, 81) #I decided that 80 would be capturing the whole dynamic
popVector[1] <- 500


for(year in seq(1, 80)){
  popVector[year+1] = popVector[year] +
    (r_d*(popVector[year])*(1- (popVector[year]/ Carrying(b0 = 0.6, b1 = Calcb1(calcRainfall((year))), d0 = 0.125) )))
}
plot(popVector, type = "o", xlab="time (yrs)", ylab="N",
    col="black", lwd=1, main="Graph of N vs. time (yrs)" )
```

Interpretation:
The population, as the rainfall was ample in the beginning years (0-20), had a carrying capacity that was increasing due to sufficient resources. However, as rainfall began to dwindle linearly, the b1 parameter would decrease (as seen in graph produced in part 1d). As b1 decreases, the carrying capacity also decreases. The carrying capacity’s shrinkage is indicative of the environment not being able to support a large populaiton, and thereby the population size decreases. The decrease is the quickest from years 30-50, and once more individuals have died out, the decrease of N slows because the environment can sustain a small batch of individuals. As there are no more rainfall, all of the population will dissipate to 0 by year 80.

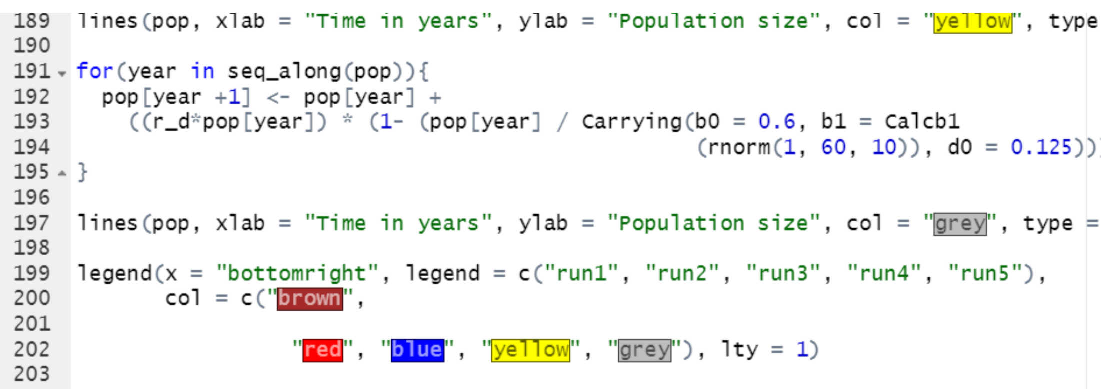
1g.


Population size
500
1000
1500
2000
0
10
20
30
40
50
Time in years
run1
run2
run3
run4
run5

Code:



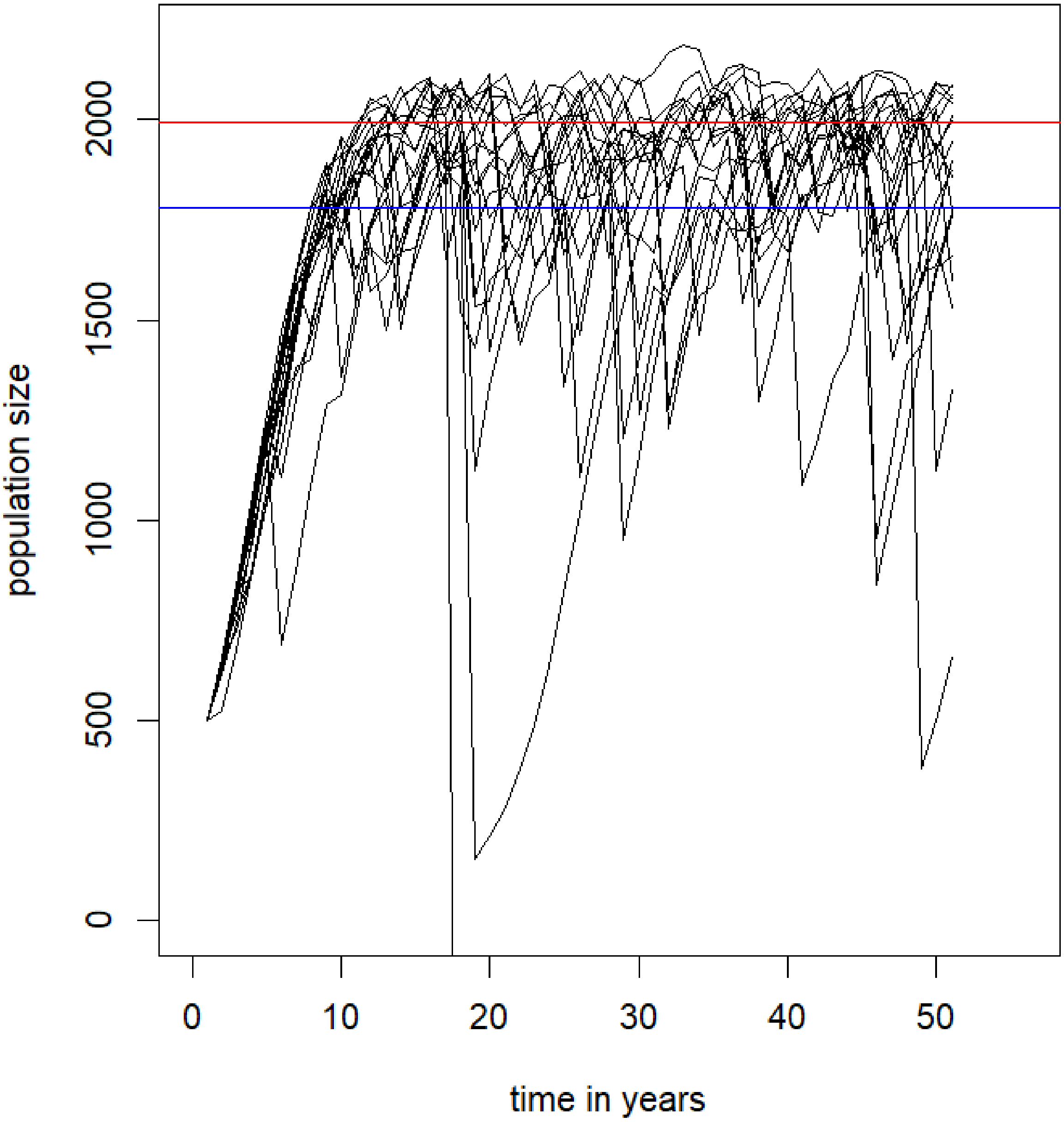
```
#------------------------revised 1g
pop <- rep(0, 51)
r_d <- 0.4
pop[1]<- 500

for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                                 (rnorm(1, 60, 10)), d0 = 0.125))))
}

plot(pop, xlab = "Time in years", ylab = "Population size", col = "brown", type = "l")


for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                                 (rnorm(1, 60, 10)), d0 = 0.125))))
}

lines(pop, xlab = "Time in years", ylab = "Population size", col = "red", type = "l")


for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                                 (rnorm(1, 60, 10)), d0 = 0.125))))
}

lines(pop, xlab = "Time in years", ylab = "Population size", col = "blue", type = "l")

for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                                 (rnorm(1, 60, 10)), d0 = 0.125))))
}

lines(pop, xlab = "Time in years", ylab = "Population size", col = "yellow", type = "l")
```



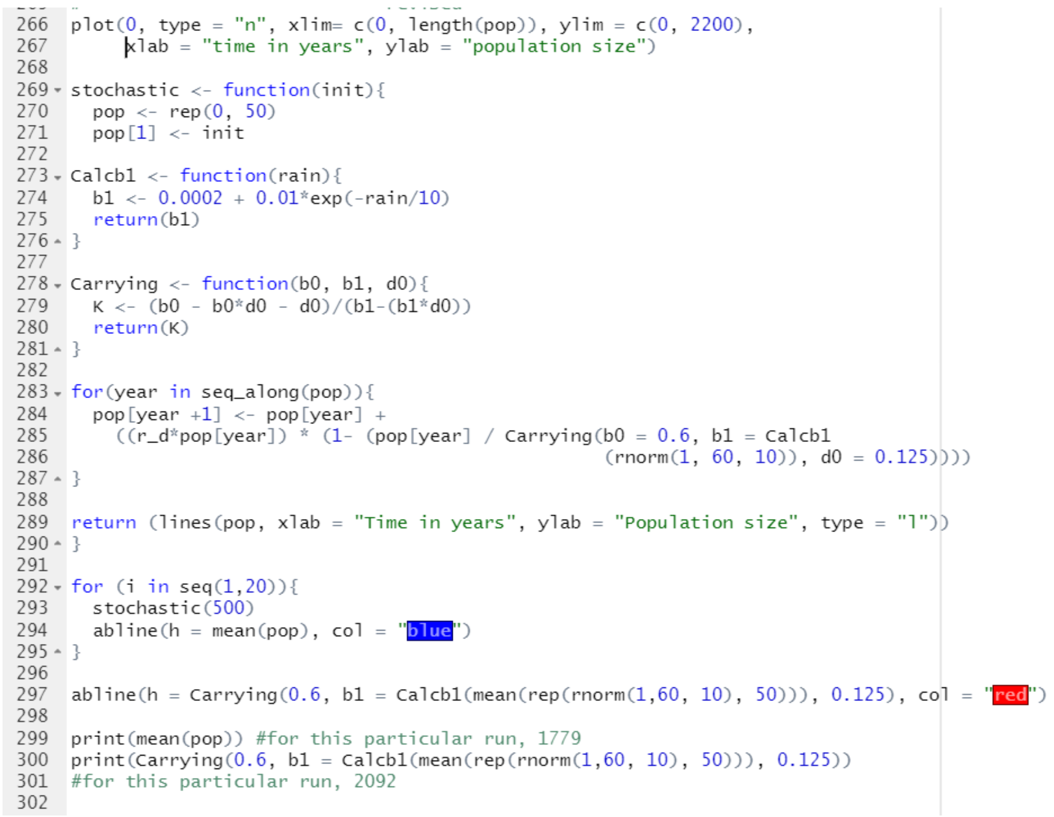
```
lines(pop, xlab = "Time in years", ylab = "Population size", col = "yellow", type = "l")

for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                                 (rnorm(1, 60, 10)), d0 = 0.125))))
}

lines(pop, xlab = "Time in years", ylab = "Population size", col = "grey", type = "l")

legend(x = "bottomright", legend = c("run1", "run2", "run3", "run4", "run5"),
       col = c("brown",

                 "red", "blue", "yellow", "grey"), lty = 1)

```

Interpretation:
The results are not the same because here the rainfall amount is pre-determined by random normal sampling and applied to the b1 calculation for every single year. In a sense, the rainfall amount is correlated directly with the carrying capacity. The long term behavior tends to N = 2000, but the nature of a stochastic model is that year to year variability is unpredictable.

1h.


population size
0
500
1000
1500
2000
0
10
20
30
40
50
time in years

Code: #1h------------------

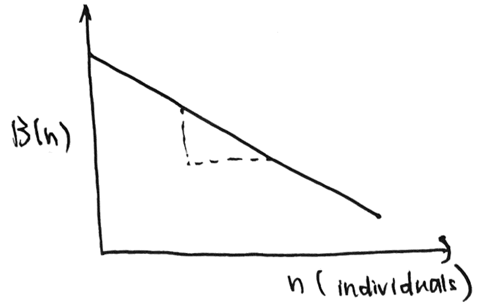


```
plot(0, type = "n", xlim= c(0, length(pop)), ylim = c(0, 2200),
     xlab = "time in years", ylab = "population size")

stochastic <- function(init){
  pop <- rep(0, 50)
  pop[1] <- init

Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                         (rnorm(1, 60, 10)), d0 = 0.125))))
}

return (lines(pop, xlab = "Time in years", ylab = "Population size", type = "l"))
}

for (i in seq(1,20)){
  stochastic(500)
  abline(h = mean(pop), col = "blue")
}

abline(h = Carrying(0.6, b1 = Calcb1(mean(rep(rnorm(1,60, 10), 50))), 0.125), col = "red")

print(mean(pop)) #for this particular run, 1779
print(Carrying(0.6, b1 = Calcb1(mean(rep(rnorm(1,60, 10), 50))), 0.125))
#for this particular run, 2092
```

Code:

```
plot(0, type = "n", xlim= c(0, length(pop)), ylim = c(0, 2200), xlab = "time in years", ylab = "population size")

stochastic <- function(init){
  pop <- rep(0, 50)
  pop[1] <- init

Calcb1 <- function(rain){
  b1 <- 0.0002 + 0.01*exp(-rain/10)
  return(b1)
}

Carrying <- function(b0, b1, d0){
  K <- (b0 - b0*d0 - d0)/(b1-(b1*d0))
  return(K)
}

for(year in seq_along(pop)){
  pop[year +1] <- pop[year] +
    ((r_d*pop[year]) * (1- (pop[year] / Carrying(b0 = 0.6, b1 = Calcb1
                                         (rnorm(1, 60, 10)), d0 = 0.125))))
}
return (lines(pop, xlab = "Time in years", ylab = "Population size", type = "l"))
}

for (i in seq(1,20)){
  stochastic(500)
  abline(h = mean(pop), col = "blue")
}

abline(h = Carrying(0.6, b1 = Calcb1(mean(rep(rnorm(1,60, 10), 50))), 0.125), col = "red")

print(mean(pop)) #for this particular run, 1779
print(Carrying(0.6, b1 = Calcb1(mean(rep(rnorm(1,60, 10), 50))), 0.125))
#for this particular run, 2092


#response: the mean population size does not match my calculated carrying
#capacity because the deterministic value is a predicted mean, whereas the
#stochastic model has varying rainfall, and is subject to change. One is
#higher and one is lower, and they shouldn't match! This is because the mean population size considers
increase in species initially but also considers the plateauing phase.
```

C

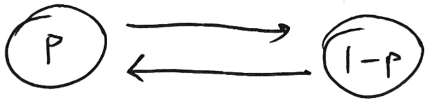


with a n of 0, then the system would be growing at its fastest rate given contribution from b_1 (birth rate)

Question 2 A. Why does colonization rate equal to $mp(1-p)$

~~They are~~ The components are p = fraction of all patches that are occupied at given time
$1-p$ = proportion that is empty.

They are multiplied together because of a bilinear relationship, When ~~all~~ there are empty patches, it is suitable to be ~~transmitted~~ transitioned to be occupied, whereas if there are more habitable patches, they are undergoing by a rate of m of transitioning to be empty. It is a bidirectionall flow of exchanging from empty to occupied, and occupied to empty.



B

The 3 simplifying assumptions are:

1. The landscape is homogeneous / the system is well-mixed
2. Interactions of species are random between patches
3. All pathes have same probability of being colonized

Leaving out:

- Local birth - death dynamics
- assuming m and e are the same for every patch, constant over time, independent of patch size, independent of distance of patch to other patches, independent of population density.
- leaving out rescue effect, do not account of external contribution
- do not account ~~of~~ for an open-system.

2c

$\frac{dP}{dt} = mp(1-p) - ep$ $\frac{dP}{dt} = 0 = mp(1-p) - ep$

$\boxed{p^* = 0}$ and $\boxed{p^* = 1 - \frac{e}{m}}$ is the equilibrium fraction of occupied patches. where:

$p^* = 1 - \frac{e}{m}$; $m \neq 0$; $m \geq e$ for $p > 0$

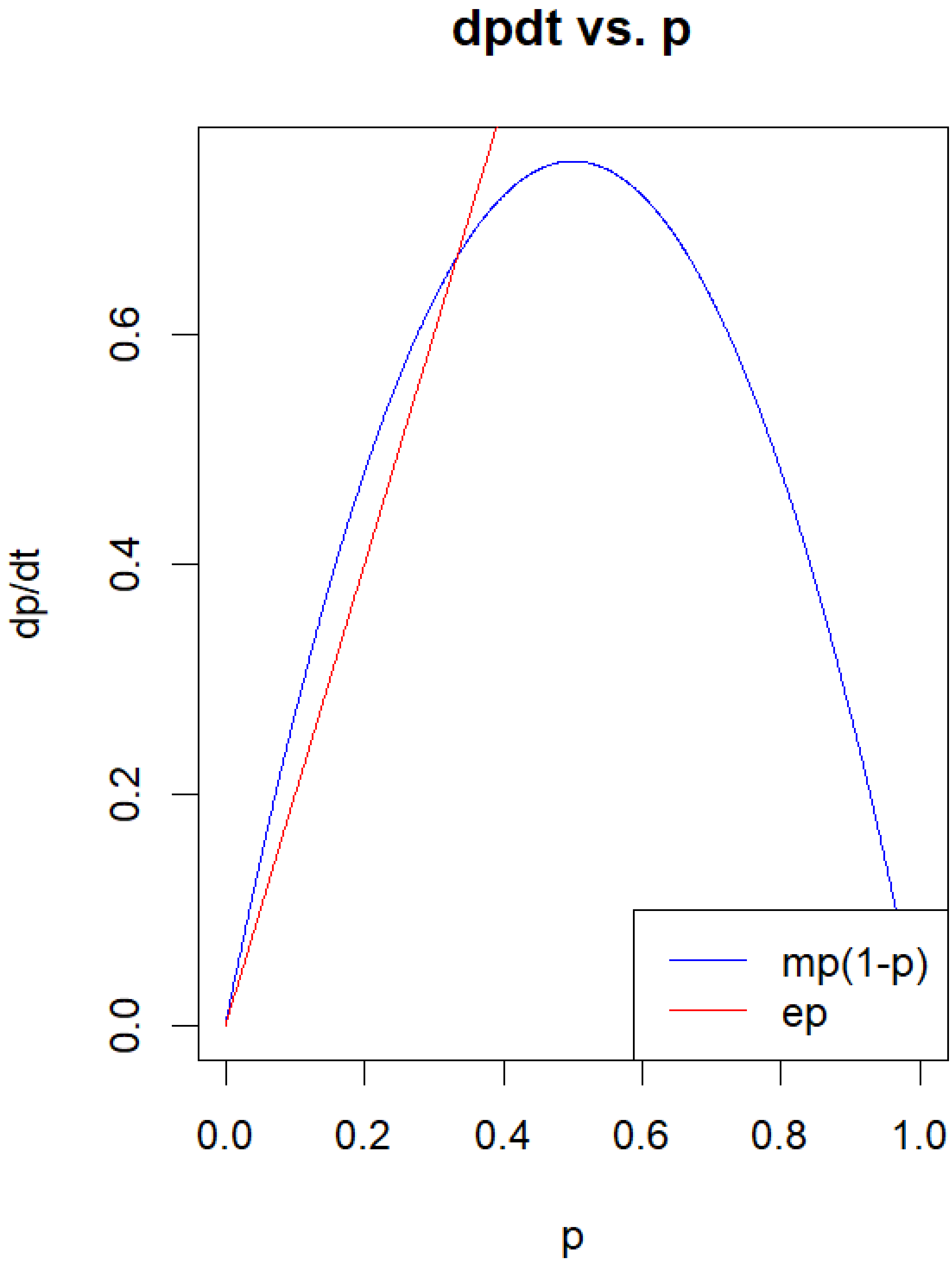
The significance of this equilibrium is that the metapopulations will persist if colonization exceeds extinction. The mathematical condition is $m \geq e$

When e is 0, all sites are occupied. So unless extinction is 0, then not all sites are occupied. This condition makes biological sense because colonization must be the dominating value to propagate the population further.

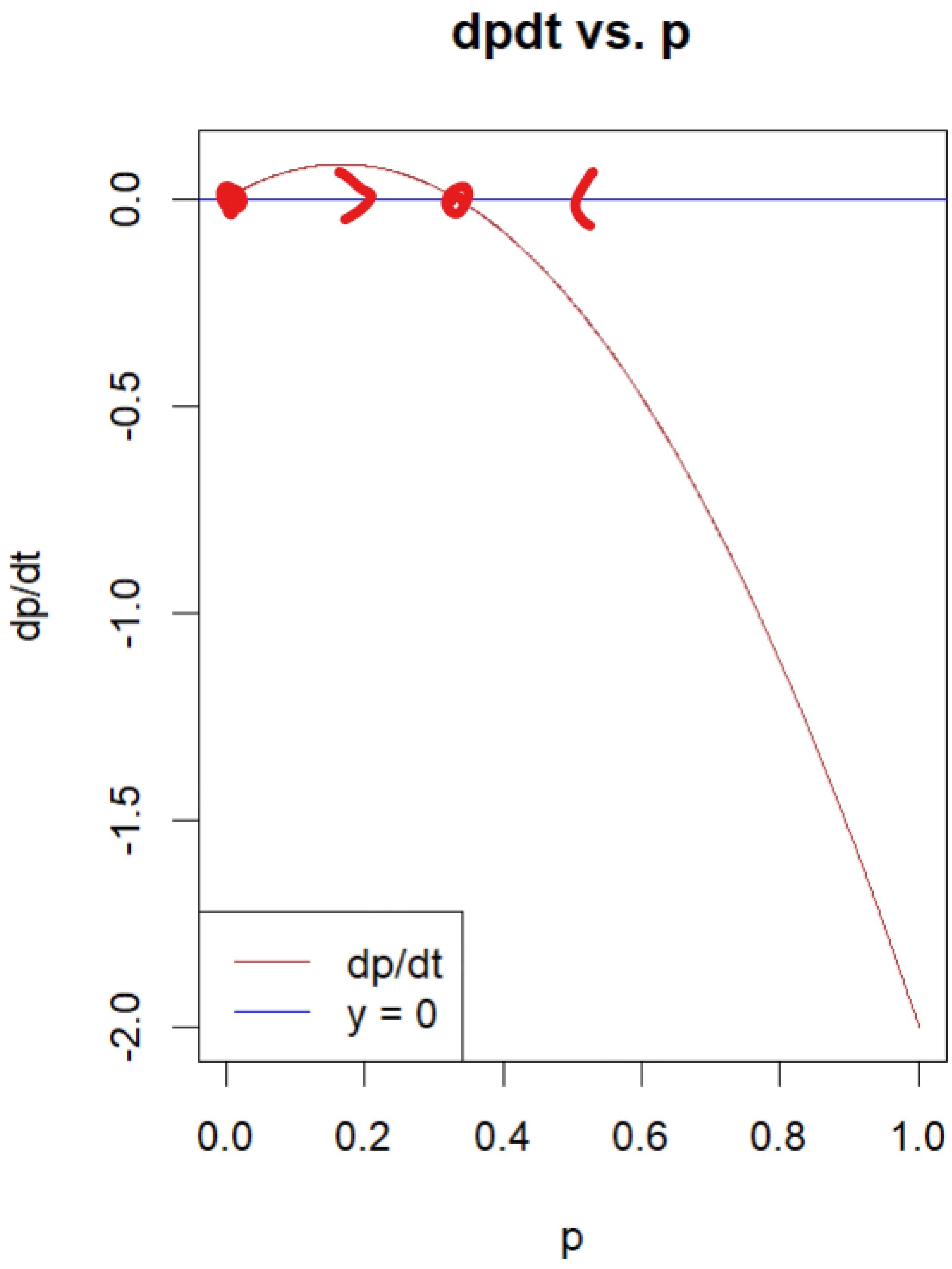
colonization greater than extinction $p^* = 1 - \frac{e}{m}$; $m > e$

second case $p^* = 0$, $e > m$

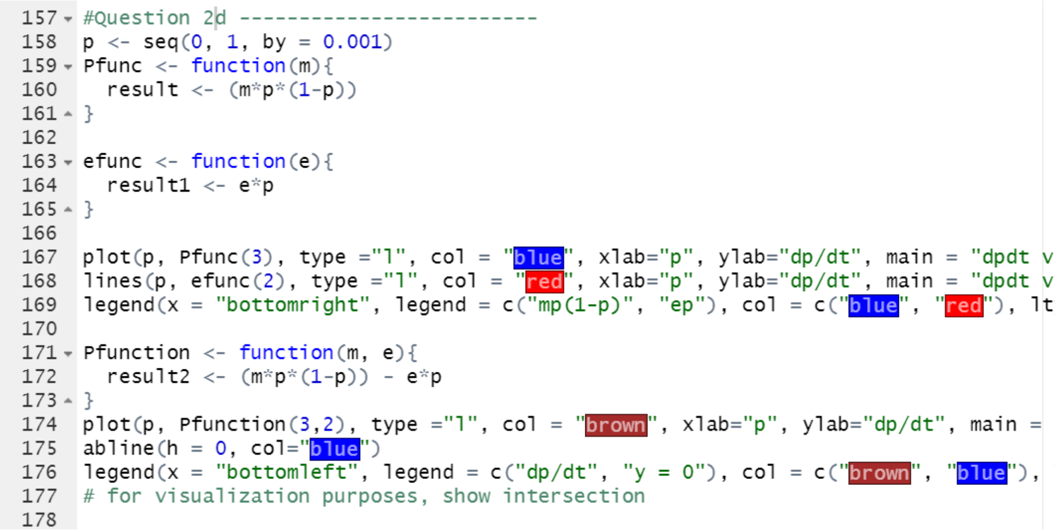
Question 2d.



Here is a plot of what happens when I separate out mp(1-p) and ep, and look for their intersection.


dpdt vs. p
dp/dt
0.0
-0.5
-1.0
-1.5
-2.0
0.0
0.2
0.4
0.6
0.8
1.0
p
dp/dt
y = 0

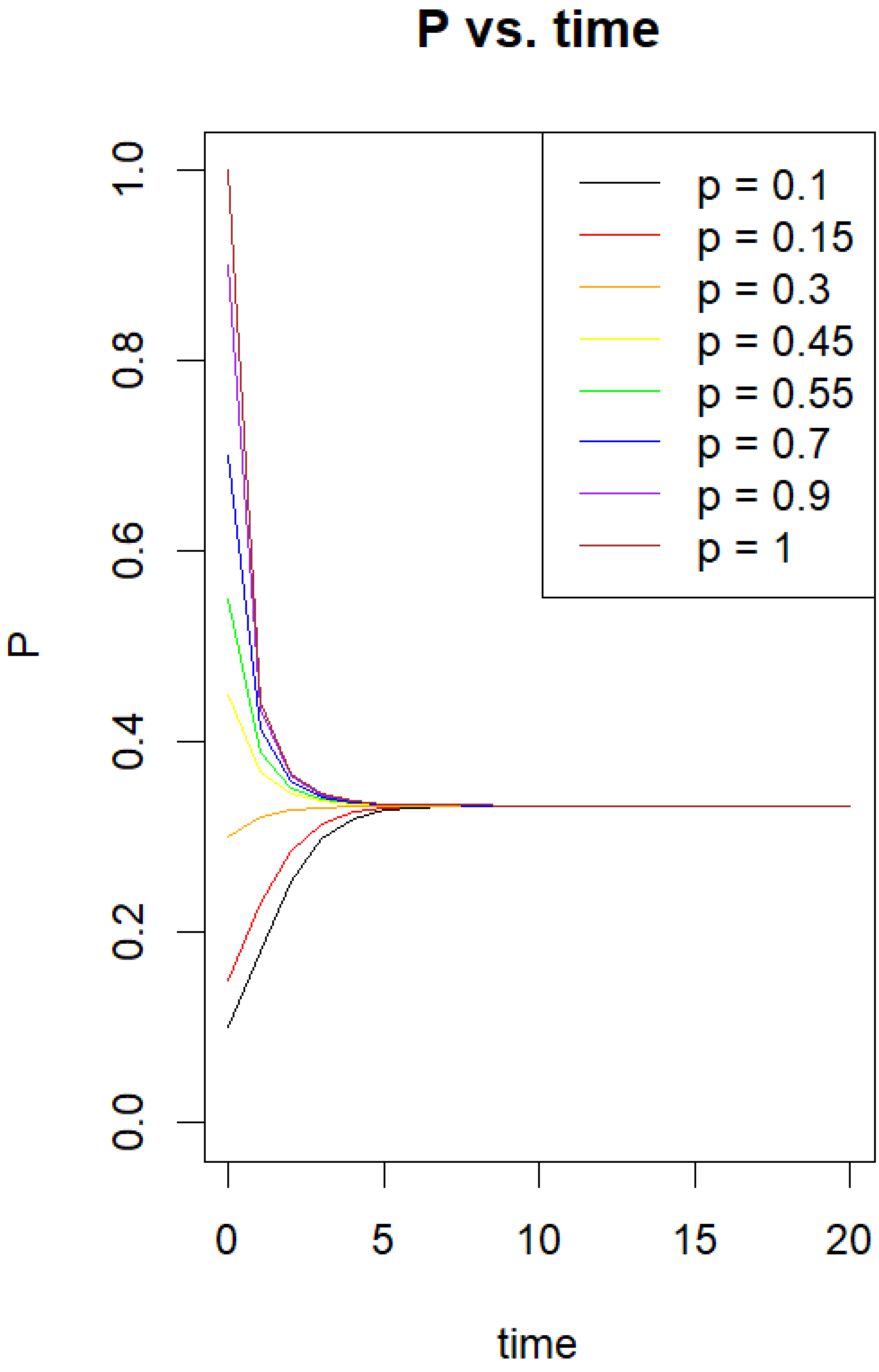
Code:



```
#Question 2d -------------------------
p <- seq(0, 1, by = 0.001)
Pfunc <- function(m){
  result <- (m*p*(1-p))
}

efunc <- function(e){
  result1 <- e*p
}

plot(p, Pfunc(3), type ="l", col = "blue", xlab="p", ylab="dp/dt", main = "dpdt vs. p")
lines(p, efunc(2), type ="l", col = "red", xlab="p", ylab="dp/dt", main = "dpdt vs. ")
legend(x = "bottomright", legend = c("mp(1-p)", "ep"), col = c("blue", "red"), lty = 1)

Pfunction <- function(m, e){
  result2 <- (m*p*(1-p)) - e*p
}
plot(p, Pfunction(3,2), type ="l", col = "brown", xlab="p", ylab="dp/dt", main = "dpdt vs. p")
abline(h = 0, col="blue")
legend(x = "bottomleft", legend = c("dp/dt", "y = 0"), col = c("brown", "blue"), lty = 1)
# for visualization purposes, show intersection

```

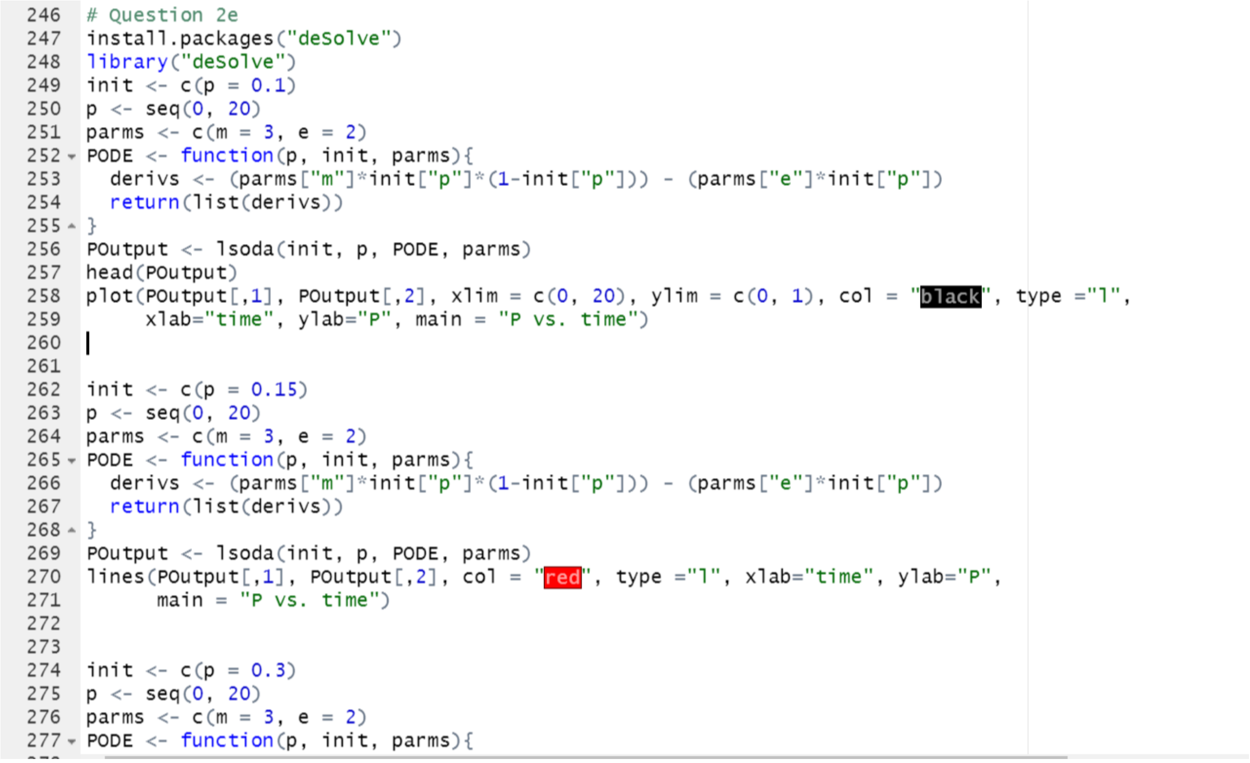
Or we can do it the traditional way, where we locate when dpdt crosses the y axis. There are two equilibrium points, one at p* = 0 and the other at around p* = ~ 0.3. The second equilibrium point is a stable equilibrium because the curve to the left is positive and the curve to the right is negative. The first equilibrium point at p* = 0 is unstable because we can't see the left curve's sign.

Question 2e.

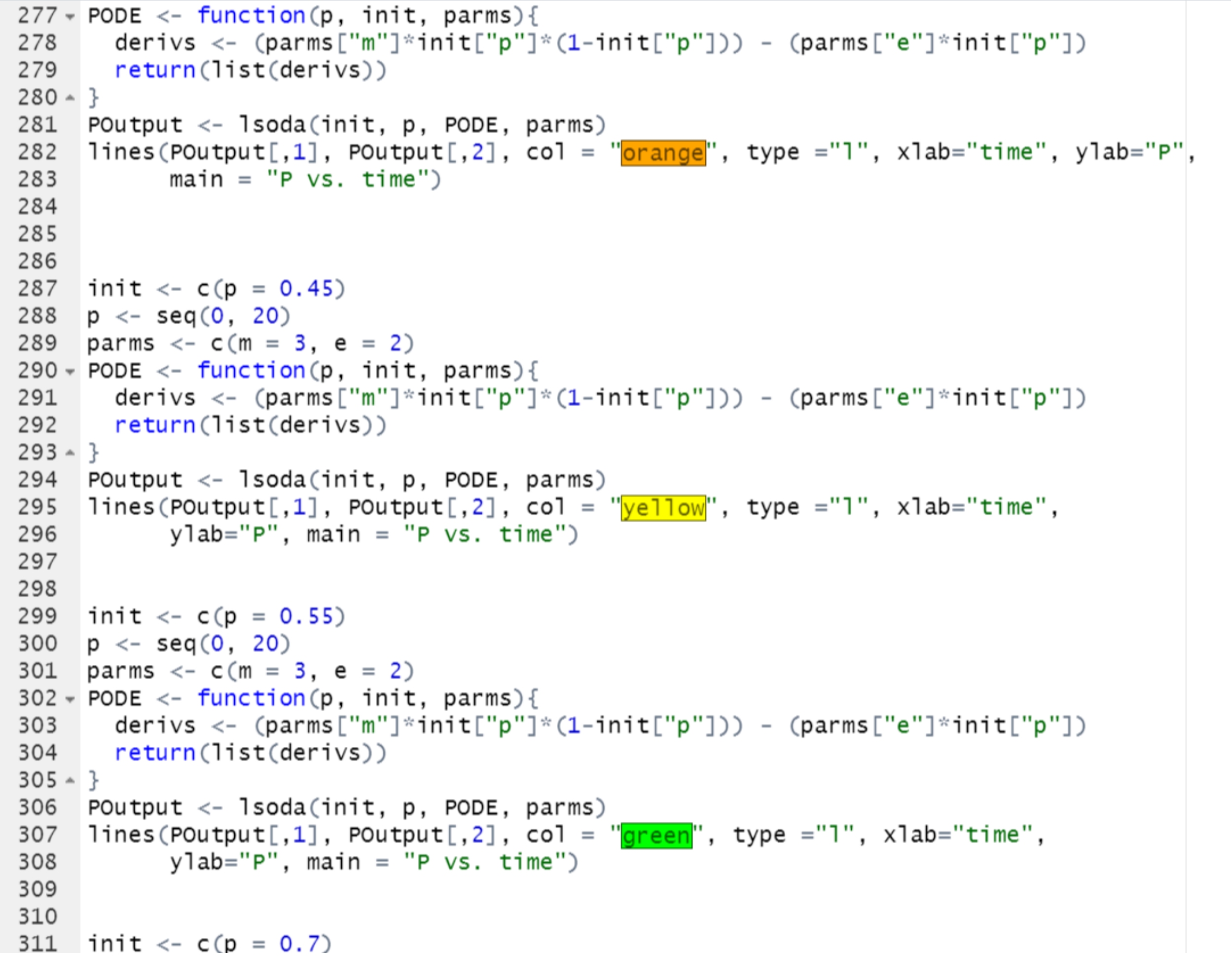

P vs. time
P
time
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
p = 0.1
p = 0.15
p = 0.3
p = 0.45
p = 0.55
p = 0.7
p = 0.9
p = 1

\

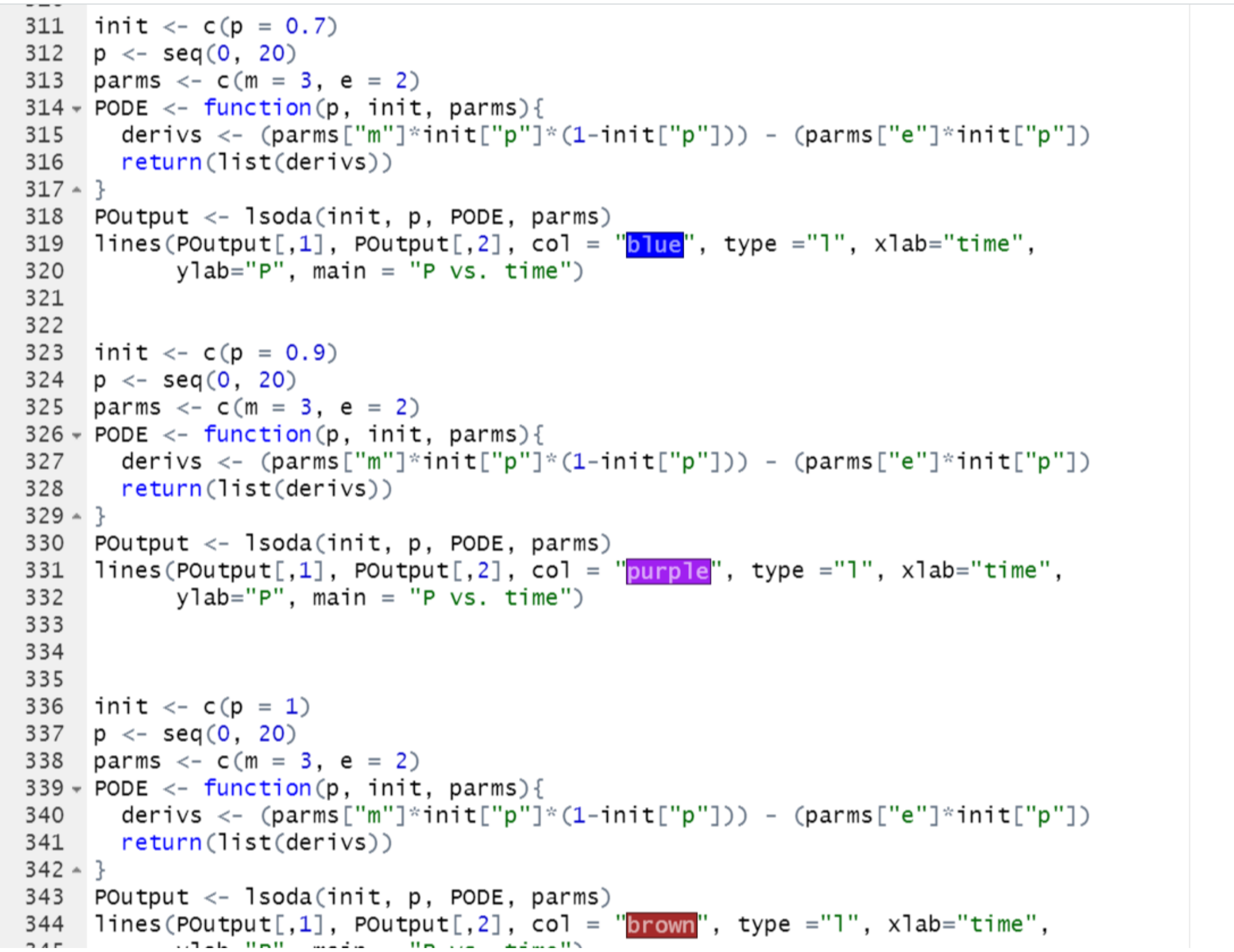
Code:



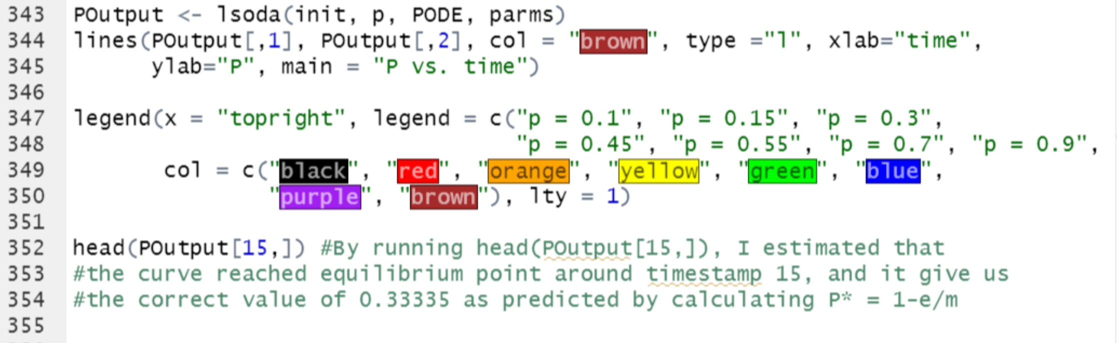
```
# Question 2e
install.packages("deSolve")
library("deSolve")
init <- c(p = 0.1)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
head(POutput)
plot(POutput[,1], POutput[,2], xlim = c(0, 20), ylim = c(0, 1), col = "black", type ="l",
     xlab="time", ylab="P", main = "P vs. time")



init <- c(p = 0.15)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "red", type ="l", xlab="time", ylab="P",
      main = "P vs. time")


init <- c(p = 0.3)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
```



```r
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "orange", type ="l", xlab="time", ylab="P",
      main = "P vs. time")



init <- c(p = 0.45)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "yellow", type ="l", xlab="time",
      ylab="P", main = "P vs. time")


init <- c(p = 0.55)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "green", type ="l", xlab="time",
      ylab="P", main = "P vs. time")


init <- c(p = 0.7)
```



```
init <- c(p = 0.7)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "blue", type ="l", xlab="time",
      ylab="P", main = "P vs. time")


init <- c(p = 0.9)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "purple", type ="l", xlab="time",
      ylab="P", main = "P vs. time")



init <- c(p = 1)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "brown", type ="l", xlab="time",
      ylab="P", main = "P vs. time")
```



```
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "brown", type ="l", xlab="time",
      ylab="P", main = "P vs. time")

legend(x = "topright", legend = c("p = 0.1", "p = 0.15", "p = 0.3",
                                  "p = 0.45", "p = 0.55", "p = 0.7", "p = 0.9", "p = 1"),
       col = c("black", "red", "orange", "yellow", "green", "blue",
               "purple", "brown"), lty = 1)

head(POutput[15,]) #By running head(POutput[15,]), I estimated that
#the curve reached equilibrium point around timestamp 15, and it give us
#the correct value of 0.33335 as predicted by calculating P* = 1-e/m
```

```
# Question 2e
install.packages("deSolve")
library("deSolve")
init <- c(p = 0.1)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
```

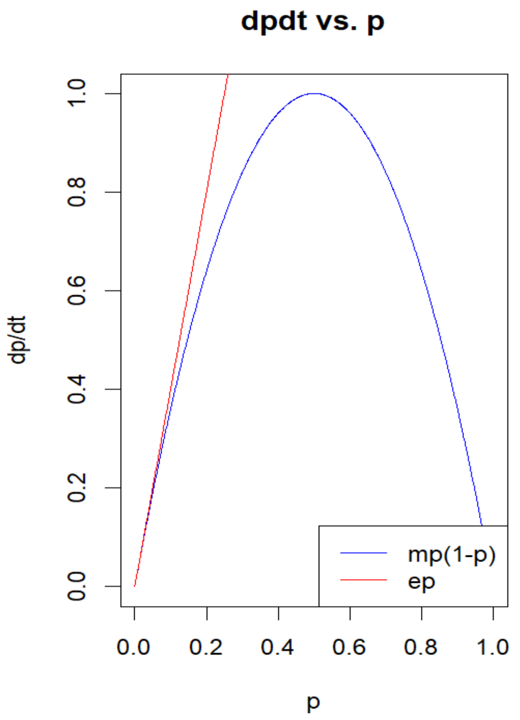
```
POutput <- lsoda(init, p, PODE, parms)
head(POutput)
plot(POutput[,1], POutput[,2], xlim = c(0, 20), ylim = c(0, 1), col = "black", type ="l",
    xlab="time", ylab="P", main = "P vs. time")


init <- c(p = 0.15)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "red", type ="l", xlab="time", ylab="P",
     main = "P vs. time")


init <- c(p = 0.3)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "orange", type ="l", xlab="time", ylab="P",
     main = "P vs. time")



init <- c(p = 0.45)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "yellow", type ="l", xlab="time",
     ylab="P", main = "P vs. time")


init <- c(p = 0.55)
```

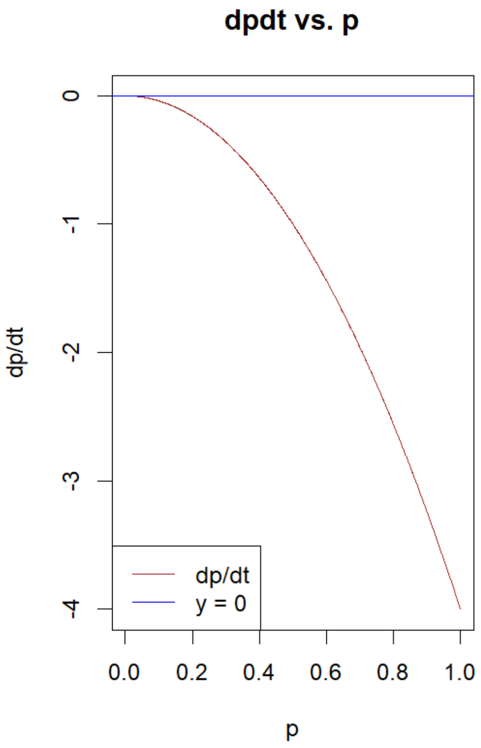
```
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "green", type ="l", xlab="time",
      ylab="P", main = "P vs. time")


init <- c(p = 0.7)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "blue", type ="l", xlab="time",
      ylab="P", main = "P vs. time")


init <- c(p = 0.9)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
lines(POutput[,1], POutput[,2], col = "purple", type ="l", xlab="time",
      ylab="P", main = "P vs. time")



init <- c(p = 1)
p <- seq(0, 20)
parms <- c(m = 3, e = 2)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsoda(init, p, PODE, parms)
```

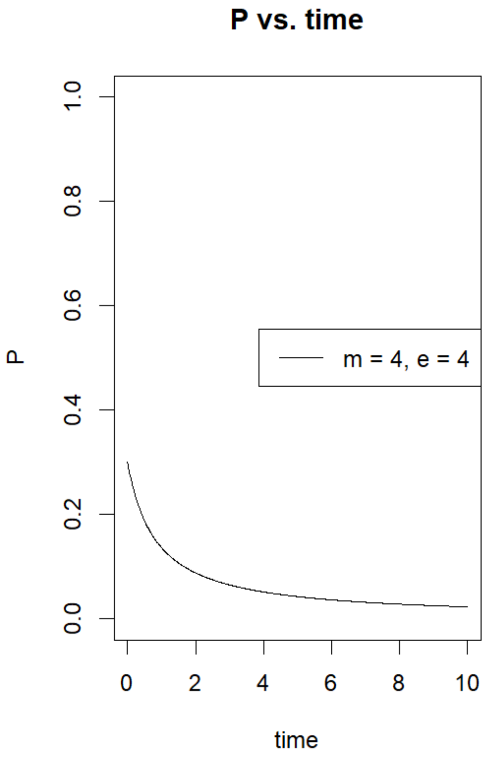
```
lines(POutput[,1], POutput[,2], col = "brown", type ="l", xlab="time",
   ylab="P", main = "P vs. time")

legend(x = "topright", legend = c("p = 0.1", "p = 0.15", "p = 0.3",
                                  "p = 0.45", "p = 0.55", "p = 0.7", "p = 0.9", "p = 1"),
    col = c("black", "red", "orange", "yellow", "green", "blue",
        "purple", "brown"), lty = 1)

head(POutput[15,]) #By running head(POutput[15,]), I estimated that
#the curve reached equilibrium point around timestamp 15, and it give us
#the correct value of 0.33335 as predicted by calculating P* = 1-e/m
```

Interpretation: I can confirm my prediction that the equilibrium of the metapopulation is located around p = 0.3. The curve has reached equilibrium around timestep 15, and the p is 0.3335.

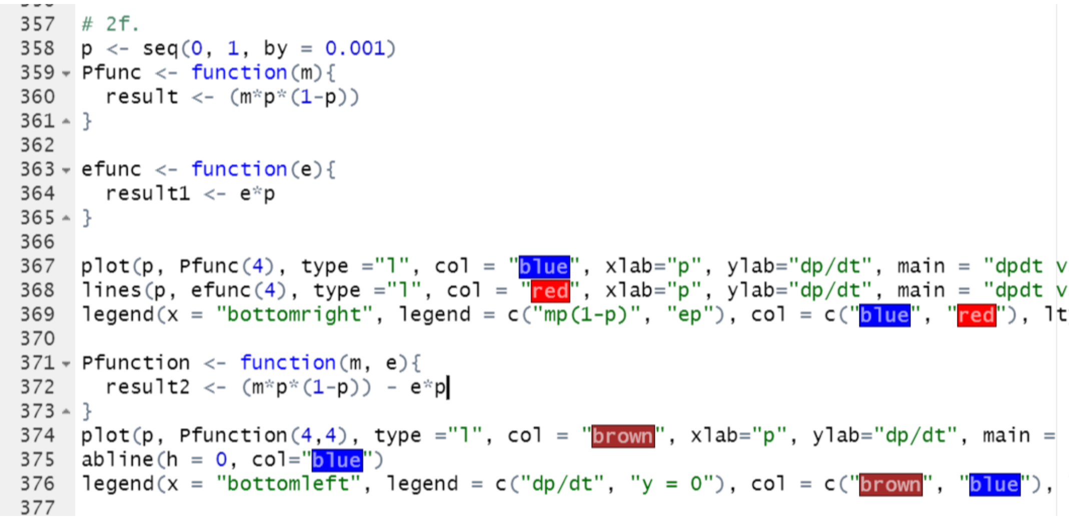
2f.


dpdt vs. p
dp/dt
0.0
0.2
0.4
0.6
0.8
1.0
p
0.0
0.2
0.4
0.6
0.8
1.0
mp(1-p)
ep

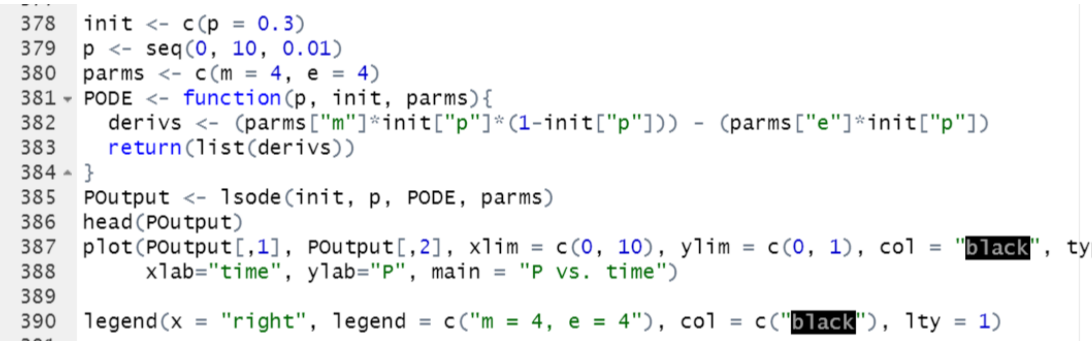

dpdt vs. p
dp/dt
0
-1
-2
-3
-4
p
0.0
0.2
0.4
0.6
0.8
1.0
dp/dt
y = 0


P vs. time
P
0.0
0.2
0.4
0.6
0.8
1.0
time
0
2
4
6
8
10
m = 4, e = 4

Code:



```
# 2f.
p <- seq(0, 1, by = 0.001)
Pfunc <- function(m){
  result <- (m*p*(1-p))
}

efunc <- function(e){
  result1 <- e*p
}

plot(p, Pfunc(4), type ="l", col = "blue", xlab="p", ylab="dp/dt", main = "dpdt vs. p")
lines(p, efunc(4), type ="l", col = "red", xlab="p", ylab="dp/dt", main = "dpdt vs. ")
legend(x = "bottomright", legend = c("mp(1-p)", "ep"), col = c("blue", "red"), lty = 1)

Pfunction <- function(m, e){
  result2 <- (m*p*(1-p)) - e*p
}
plot(p, Pfunction(4,4), type ="l", col = "brown", xlab="p", ylab="dp/dt", main = "dpdt vs. p")
abline(h = 0, col="blue")
legend(x = "bottomleft", legend = c("dp/dt", "y = 0"), col = c("brown", "blue"), lty = 1)

```



```
init <- c(p = 0.3)
p <- seq(0, 10, 0.01)
parms <- c(m = 4, e = 4)
PODE <- function(p, init, parms){
  derivs <- (parms["m"]*init["p"]*(1-init["p"])) - (parms["e"]*init["p"])
  return(list(derivs))
}
POutput <- lsode(init, p, PODE, parms)
head(POutput)
plot(POutput[,1], POutput[,2], xlim = c(0, 10), ylim = c(0, 1), col = "black", type ="l",
     xlab="time", ylab="P", main = "P vs. time")

legend(x = "right", legend = c("m = 4, e = 4"), col = c("black"), lty = 1)
```

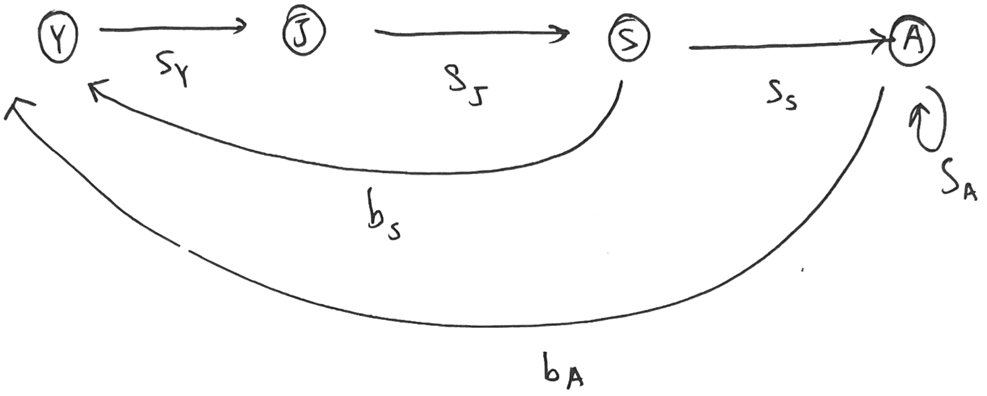
Interpretation: for m = 4 and e = 4, the dp/dt vs. p graph shows that there is only one unstable equilibrium point at p* = 0. In the P vs. time graph, there is a change in the stability convergence trend, where p tends to 0 instead of 0.333 like previously with m = 3 and e = 2. We can conclude that the behavior is different.

2g. The model is related to the continuous-time logistic growth model by both concerning the persistence of the species with time. Where the logistic growth model is considering the dynamics of carrying capacity and population size, the metapopulation sees patch availability proportion and how this availability of land can influence colonization and extinction rate. Both dynamics, if picking the right parameters, will have equilibrium points.

Question 3

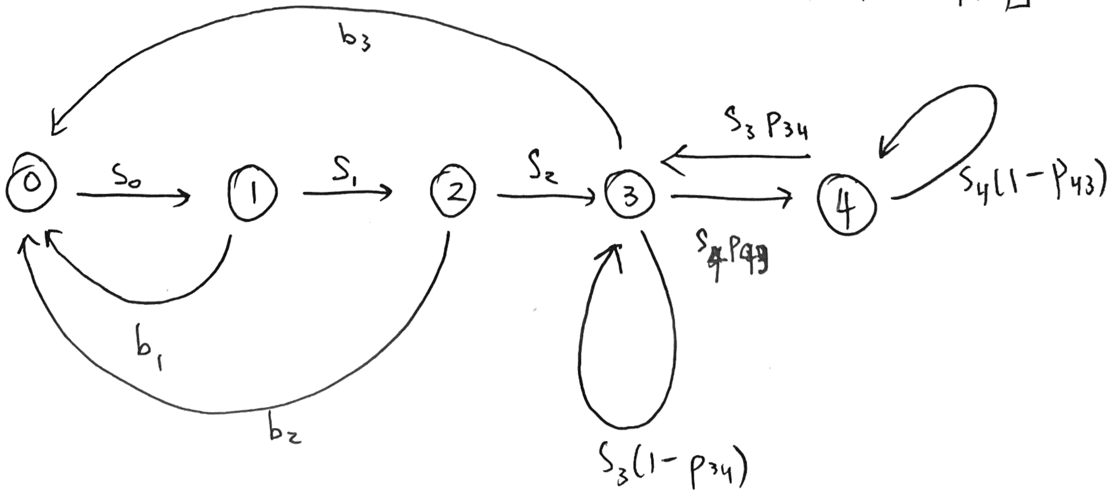
A

$$L = \begin{bmatrix} 0 & 0 & b_S & b_A \\ S_Y & 0 & 0 & 0 \\ 0 & S_J & 0 & 0 \\ 0 & 0 & S_S & S_A \end{bmatrix}$$



B

$$L = \begin{bmatrix} 0 & b_1 & b_2 & b_3 & 0 \\ S_0 & 0 & 0 & 0 & 0 \\ 0 & S_1 & 0 & 0 & 0 \\ 0 & 0 & S_2 & S_3(1-p_{34}) & S_4 P_{43} \\ 0 & 0 & 0 & S_3 p_{34} & S_4(1-p_{43}) \end{bmatrix}$$



Question 3C.

$$L = \begin{bmatrix} 0 & 0 & b & b & b & b & b \\ S_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & S_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_5 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & 0 & b \\ S_0 & 0 & 0 \\ 0 & S_1 & 0 \end{bmatrix}$$

The stage-structured matrix is not an exact match because we have reduced the specificity of the parameters in the age-structured model, the b_2 b_3, b_4, b_5, b_6 are condensed into b_2, which is nonspecific.

3D

$$L = \begin{array}{ccccccc|c}
0 & 1 & 2 & 3 & 4 & 5 & 6 & \\
\hline
0 & 0 & 0 & 0 & 0 & b_5 & b_6 & 0 \\
p_{01} & 0 & 0 & 0 & 0 & 0 & c_6 & 1 \\
p_{02} & p_{12} & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & p_{13} & p_{23} & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 0 & p_{34} & 0 & 0 & 0 & 4 \\
0 & 0 & 0 & p_{35} & 0 & s_5 & p_{65} & 5 \\
0 & 0 & 0 & p_{36} & p_{46} & p_{56} & s_6 & 6
\end{array}$$