Obermayer_Assignment8

Alyssa Obermayer

12/2/2020

Packages

```
# Load required packages
library(plyr)
```

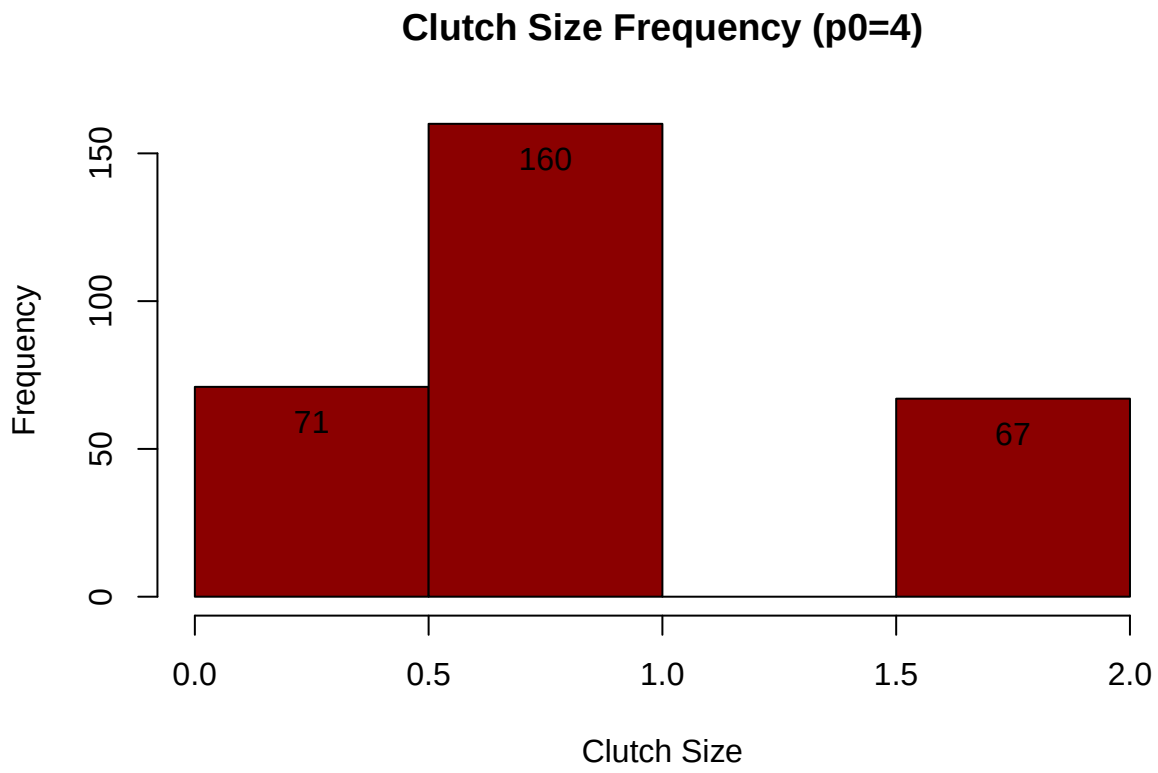
Question 1

Part 1

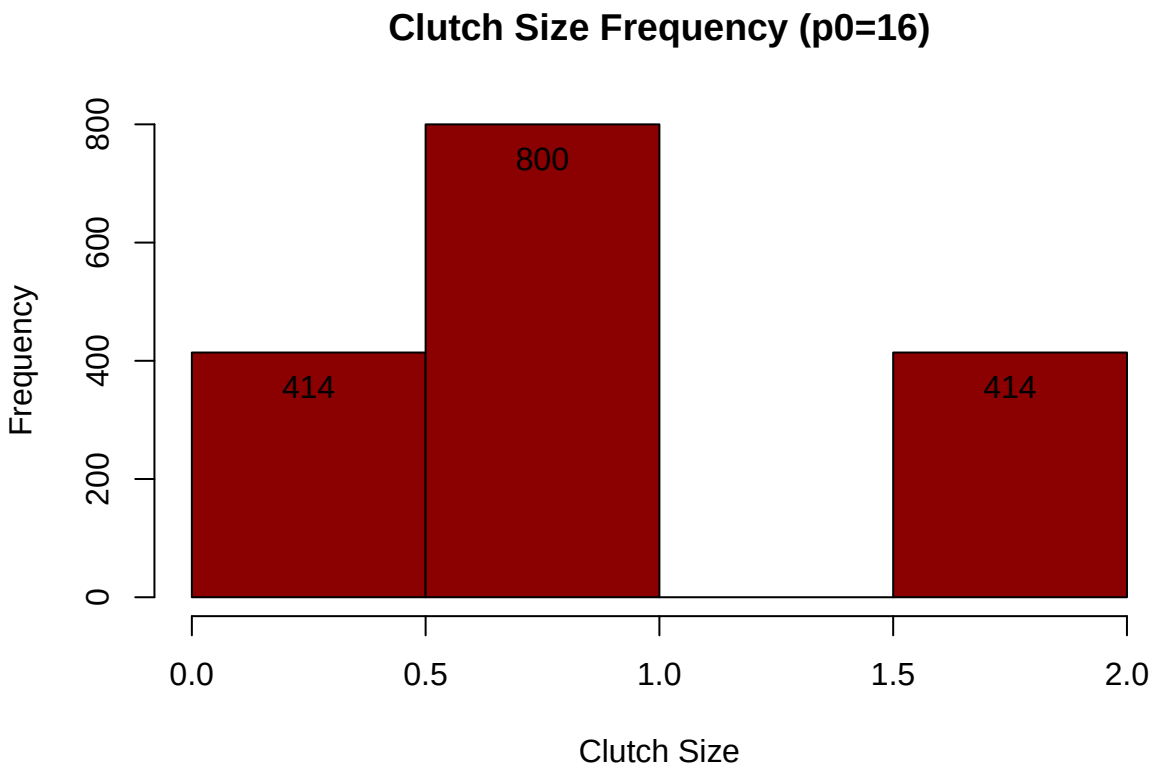
To start we took a look at the demographic stochasticity function that we used previously to determine population size over time stochastically. This function already has the basic elements to determine clutch size, so to see how that varies in the population we just need to create a list to hold the various values that we receive from each run through for the population. This list was then generated for each starting population to see the distribution of clutch sizes over a 100 year period.

```
clutch <- c()
dem.stochc <- function(p0, p1, n0, times) {
    n = n0
    for (i in 1:times) {
        new_n = 0
        if (n[i] == 0) {
            n <- c(n, new_n)
        } else {
            for (j in 1:n[i]) {
                luck = runif(1)
                if (luck < p0) {
                  new_n = new_n
                  clutch <<- append(clutch, 0)
                } else {
                  if (luck < p0 + p1) {
                    new_n = new_n + 1
                    clutch <<- append(clutch, 1)
                  } else {
                    new_n = new_n + 2
                    clutch <<- append(clutch, 2)
                  }
                }
            }
```

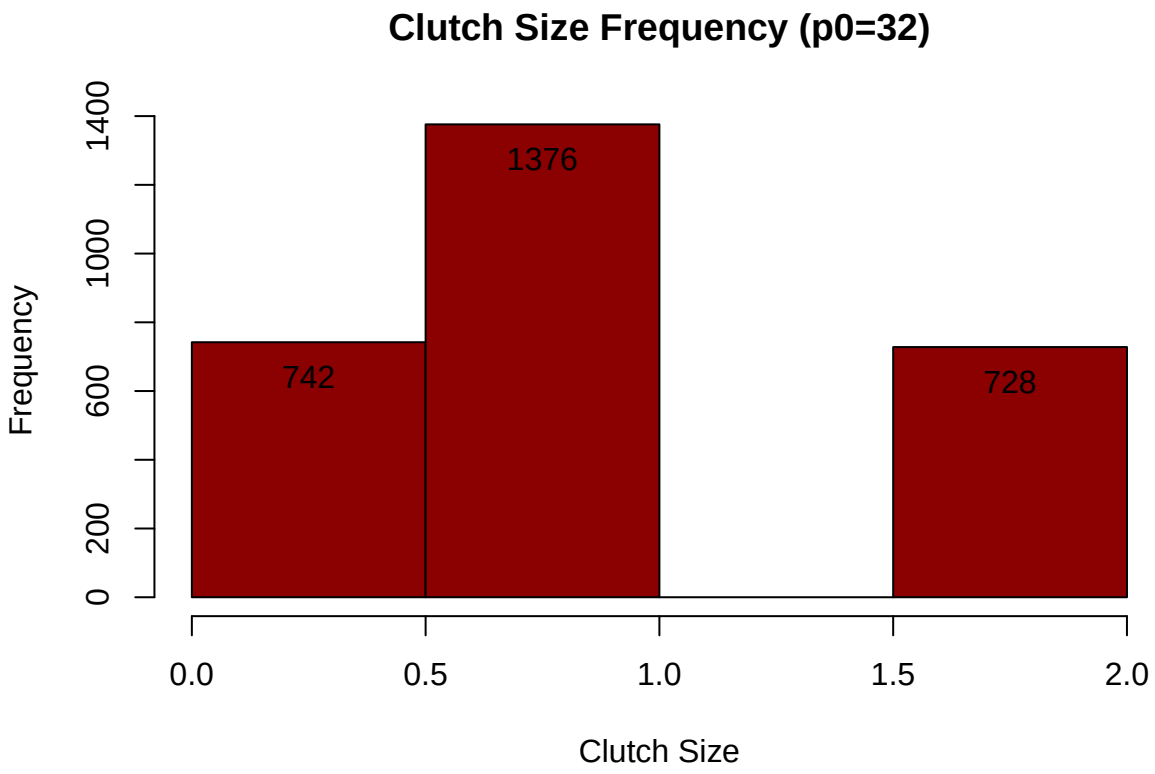
```
            n <- c(n, new_n)
        }
    }
    return(n)
}

clutch <- c()
set.seed(1)
p4 <- dem.stochc(0.25, 0.5, 4, 100)
h4 <- hist(clutch, breaks = 3, xlab = "Clutch Size", main = "Clutch Size Frequency (p0=4)",
    col = "red4")
text(h4$mids, h4$counts, labels = h4$counts, adj = c(0.5, 2))
```



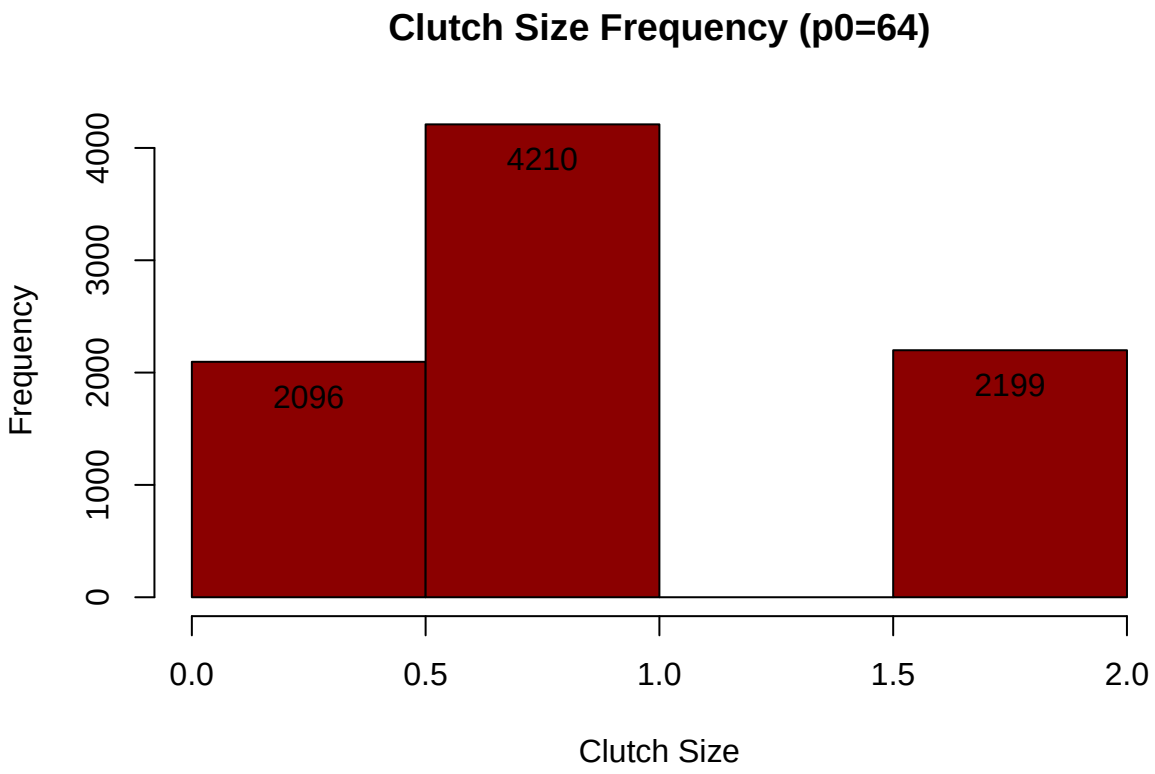

```
clutch <- c()
set.seed(1)
p16 <- dem.stochc(0.25, 0.5, 16, 100)
h16 <- hist(clutch, breaks = 3, xlab = "Clutch Size", main = "Clutch Size Frequency (p0=16)",
    col = "red4")
text(h16$mids, h16$counts, labels = h16$counts, adj = c(0.5,
    2))
```



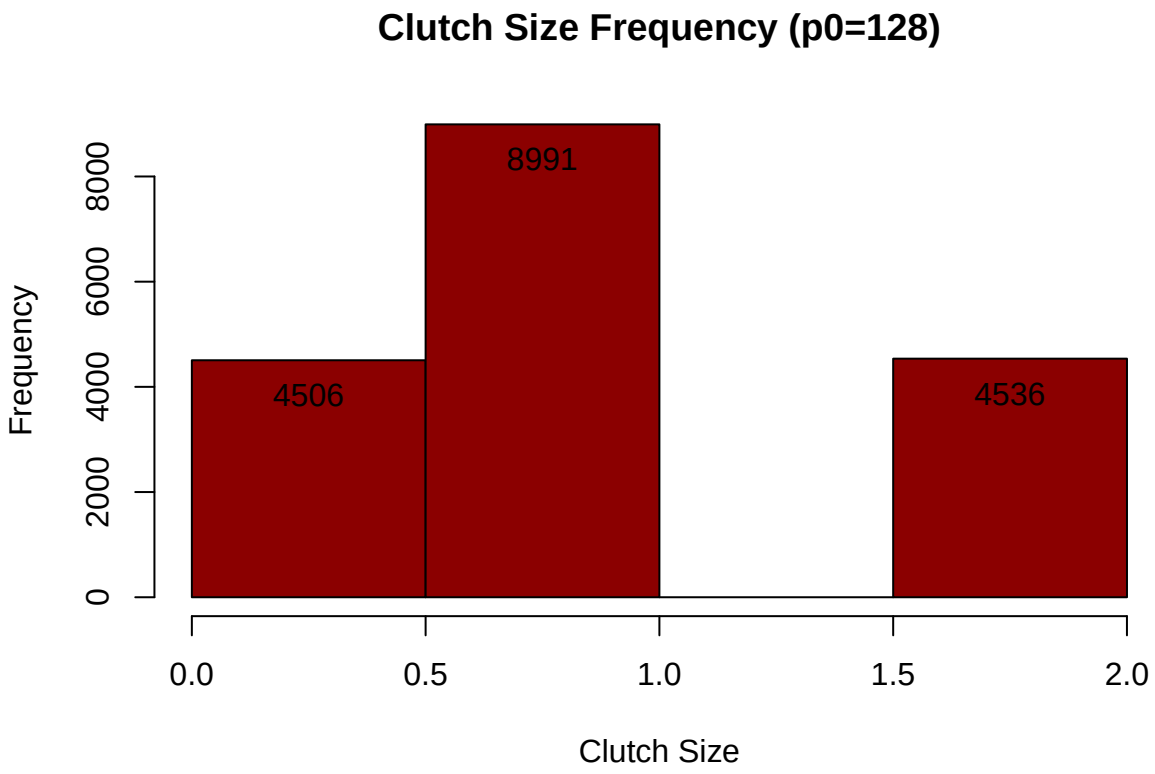

```
clutch <- c()
set.seed(1)
p32 <- dem.stochc(0.25, 0.5, 32, 100)
h32 <- hist(clutch, breaks = 3, xlab = "Clutch Size", main = "Clutch Size Frequency (p0=32)",
    col = "red4")
text(h32$mids, h32$counts, labels = h32$counts, adj = c(0.5,
    2))
```




```
clutch <- c()
set.seed(1)
p64 <- dem.stochc(0.25, 0.5, 64, 100)
h64 <- hist(clutch, breaks = 3, xlab = "Clutch Size", main = "Clutch Size Frequency (p0=64)",
    col = "red4")
text(h64$mids, h64$counts, labels = h64$counts, adj = c(0.5,
    2))
```




```
clutch <- c()
set.seed(1)
p128 <- dem.stochc(0.25, 0.5, 128, 100)
h128 <- hist(clutch, breaks = 3, xlab = "Clutch Size", main = "Clutch Size Frequency (p0=128)",
    col = "red4")
text(h128$mids, h128$counts, labels = h128$counts, adj = c(0.5,
    2))
```

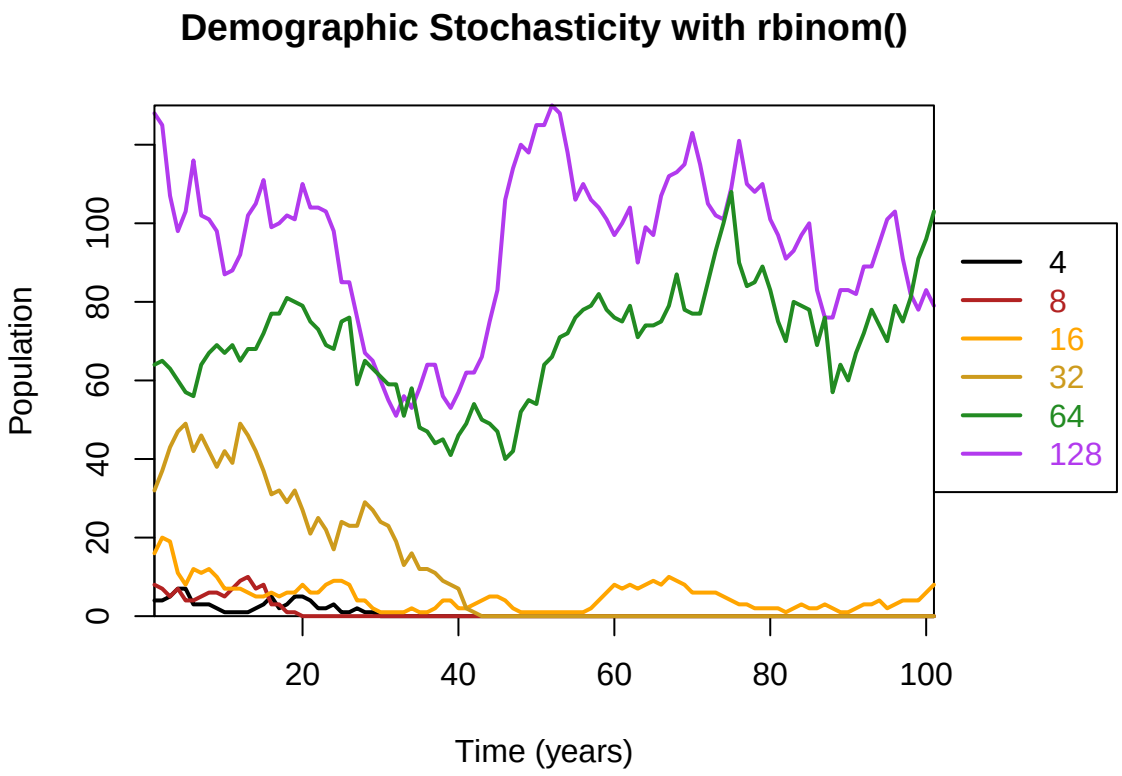


Part 2

Furthermore, when stochastically modeling a population over time there are various terms of probability distributions that can be used to represent a similar affect. Previously, we used a distribution determined by "runif(1)" which would generate a random number between 0 and 1 and go off of the given probability to generate a certain number of offspring. Below, I adjusted the previous function to run with a binomial probability distribution. For this I generated a random number 0 or 1 with a 0.5 probability of each and if that gave me a 1, there would be 1 offspring, but if I got a 0, it would go on to replicate that binomial function a second time. This second binomial would be similar to the first, though, if I received a 0, there would be 0 offspring, but if I received a 1 there would be 2 offspring. Together, this demonstrated a model similar to the runif() model from previous lessons.

```
dem.stoch <- function(n0, times) {
    n = n0
    for (i in 1:times) {
        new_n = 0
        if (n[i] == 0) {
            n <- c(n, new_n)
        } else {
            for (j in 1:n[i]) {
                luck = rbinom(n = 1, size = 1, p = 0.5)
                if (luck == 1) {
                  new_n = new_n + 1
                } else {
                  nluck = rbinom(n = 1, size = 1, p = 0.5)
```

```
                if (nluck == 0) {
                  new_n = new_n + 0
                } else {
                  new_n = new_n + 2
                }
              }
            }
            n <- c(n, new_n)
        }
    }
    return(n)
}

set.seed(1)
par(xpd = T, mar = par()$mar + c(0, 0, 0, 6))
n0.128 <- dem.stoch(128, 100)
plot(n0.128, ylim = c(0, max(n0.128)), type = "l", ylab = "Population",
    xlab = "Time (years)", main = "Demographic Stochasticity with rbinom()",
    lwd = 2, col = "darkorchid2", xaxs = "i", yaxs = "i")
lines(dem.stoch(4, 100), ylab = "population", xlab = "time",
    lwd = 2, col = "black")
lines(dem.stoch(8, 100), ylab = "population", xlab = "time",
    lwd = 2, col = "firebrick")
lines(dem.stoch(16, 100), ylab = "population", xlab = "time",
    lwd = 2, col = "orange")
lines(dem.stoch(32, 100), ylab = "population", xlab = "time",
    lwd = 2, col = "goldenrod3")
lines(dem.stoch(64, 100), ylab = "population", xlab = "time",
    lwd = 2, col = "forestgreen")
legend(legend = c("4", "8", "16", "32", "64", "128"), text.col = c("black",
    "firebrick", "orange", "goldenrod3", "forestgreen", "darkorchid2"),
    lty = 1, col = c("black", "firebrick", "orange", "goldenrod3",
        "forestgreen", "darkorchid2"), lwd = 2, x = 101, y = 100)
```



Question 2

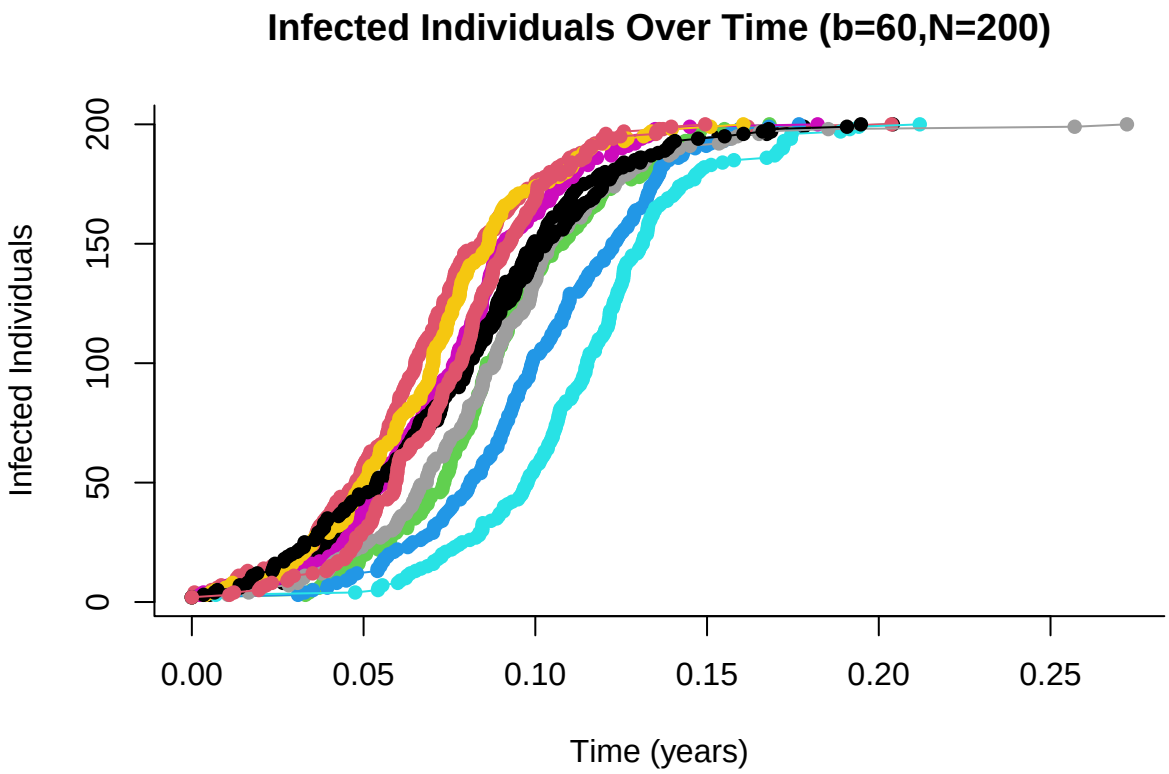
Part 1

The model that is being implemented seems to represent frequency dependent transmission. This occurs when the per capita contact rate between susceptible and infected individuals is independent of the population density. In the models below, as population increases and the transmission parameter (beta) stays the same, the curve of infected individuals on the graph does not become steeper, indicating that the higher risk of transmission does not causing individuals to become infected at a faster rate with a larger population. Additionally, the population size parameter (N) is usually canceled out of the denominator in the density dependent rate of infection which does not seem to be the case here.

Part 2

Below are the main functions and a run of the code, taken from the text, which I will use to produce multiple SI plots with varying parameters. I decided to take literature regarding the sexually transmitted disease, syphilis from Feldman and Mishra (2019) who included a range of transmission probability per sexual act from 0.09-0.64 which I will model with varying populations as well. When it comes to changing the transmission type, it should depend on the type of disease you are modeling and how that disease is transmitted. In this case I would stick with frequency dependent transmission because I am modeling an STD which can depend on contact rate but that factor is independent of population size itself. There is an assumed sense of mixing and a heterogeneous nature to the population one may come in contact with.

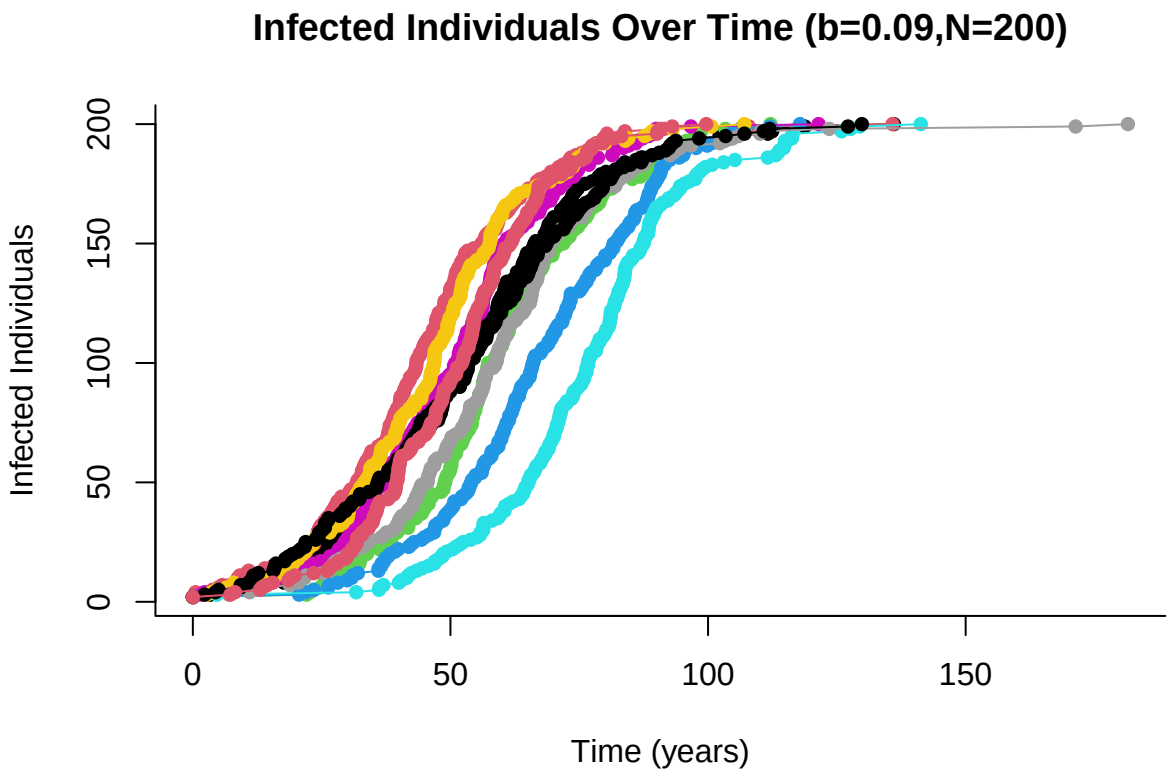
```r
SI.simul <- function(x, params, nstep) {
    ## set up an array to store results
    output <- array(dim = c(nstep + 1, 3))
    ## name the variables in the array
    colnames(output) <- c("time", "X", "Y")
    output[1, ] <- x  # initial condition
    ## iterate the model for nstep events
    for (k in 1:nstep) {
        ## update x and store result
        output[k + 1, ] <- x <- SI.onestep(x, params)
    }
    as.data.frame(output)
}

SI.onestep <- function(x, params) {
    ## the second element of x is number of susceptibles X
    X <- x[2]
    ## the third element of x is number of infecteds Y
    Y <- x[3]
    event.rate <- params["beta"] * X * Y/(X + Y)
    ## how much time do we wait for this event?
    tau <- rexp(n = 1, rate = event.rate)
    c(tau = x[1] + tau, X = X - 1, Y = Y + 1)
}

set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 200  # size of the population #CHANGING THIS PARAMETER
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 60, gamma = 365/13)  # parameters (R0=2.1) #CHANGING BETA PARAMETER
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=60,N=200)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



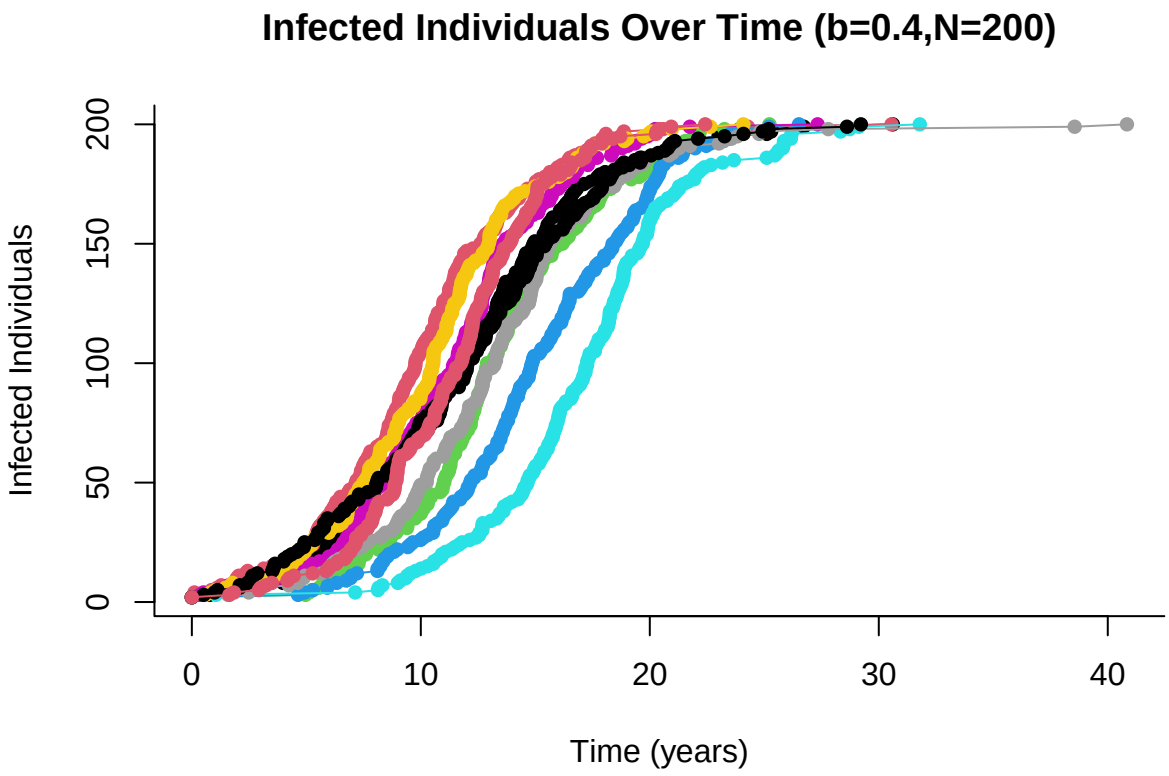
To start I will keep the population size the same as the reference (N=200) and produce models with varying beta values. When observing the models below they all show a similar logistic curve because they are leveling off when the entire population has become infected because there is no recovery, as well as this is a closed population. The main difference to see is that as the transmission parameter (beta) increases in this constant population, the amount of time for the population to become infected decreases, with some models at b=0.09 taking over 150 years to infect and some where b=0.64 taking just over 25 years.

```
# beta = 0.09
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 200  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.09, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.09,N=200)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
```

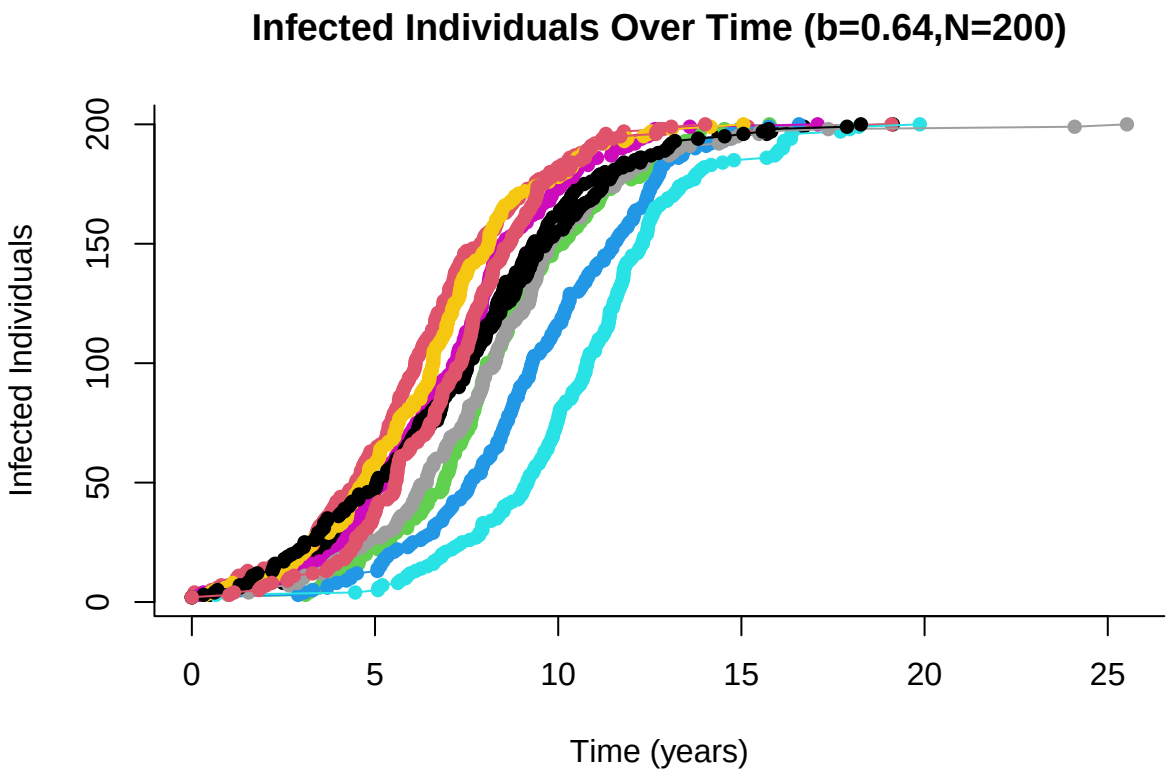
```
    type = "o", pch = 16)
```




```
# beta = 0.4
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 200  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.4, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.4,N=200)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



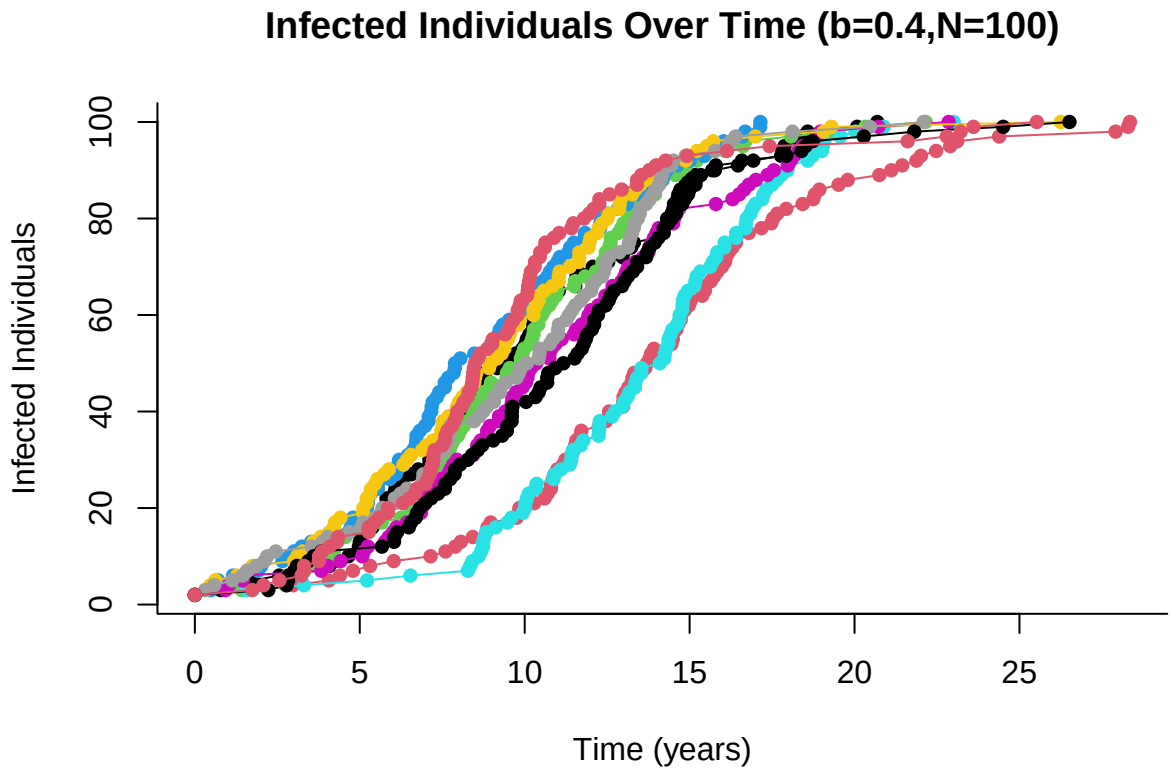

```r
# beta = 0.64
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 200  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.64, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.64,N=200)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



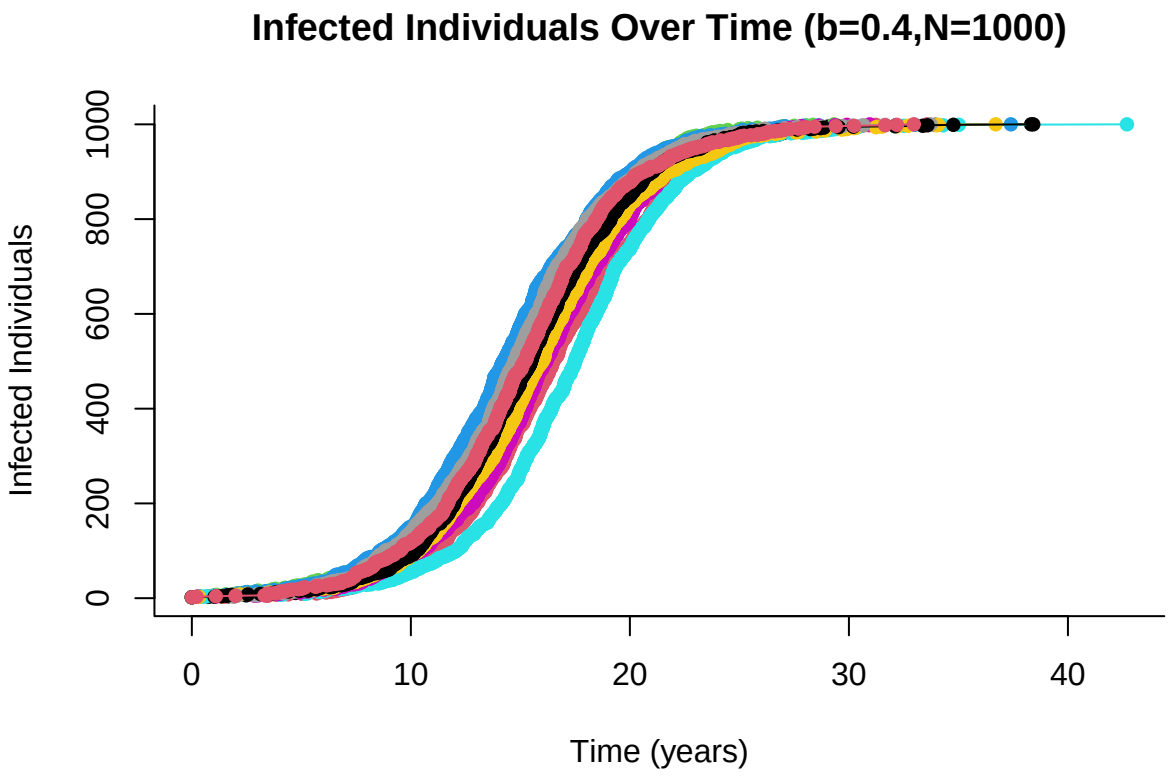
Next I will show various models with the infection rate staying constant but with an increase in population size. Just as above, there is a similar curve to the model with my beta staying constant and population increasing. I started the population just below the model from the given literature (King, 2017) and increased it to N=1000 and then N=4000. With this relatively drastic increase in population size the time for this infection to overcome the population does not change drastically with it. It does vary from just over 25 years in some cases at N=100 to taking just under 50 years in a population of N=4000. When compared to constant population and increasing beta this change in time scale is relatively minimal.

```
# pop.size = 100
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 100  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.4, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.4,N=100)")
```

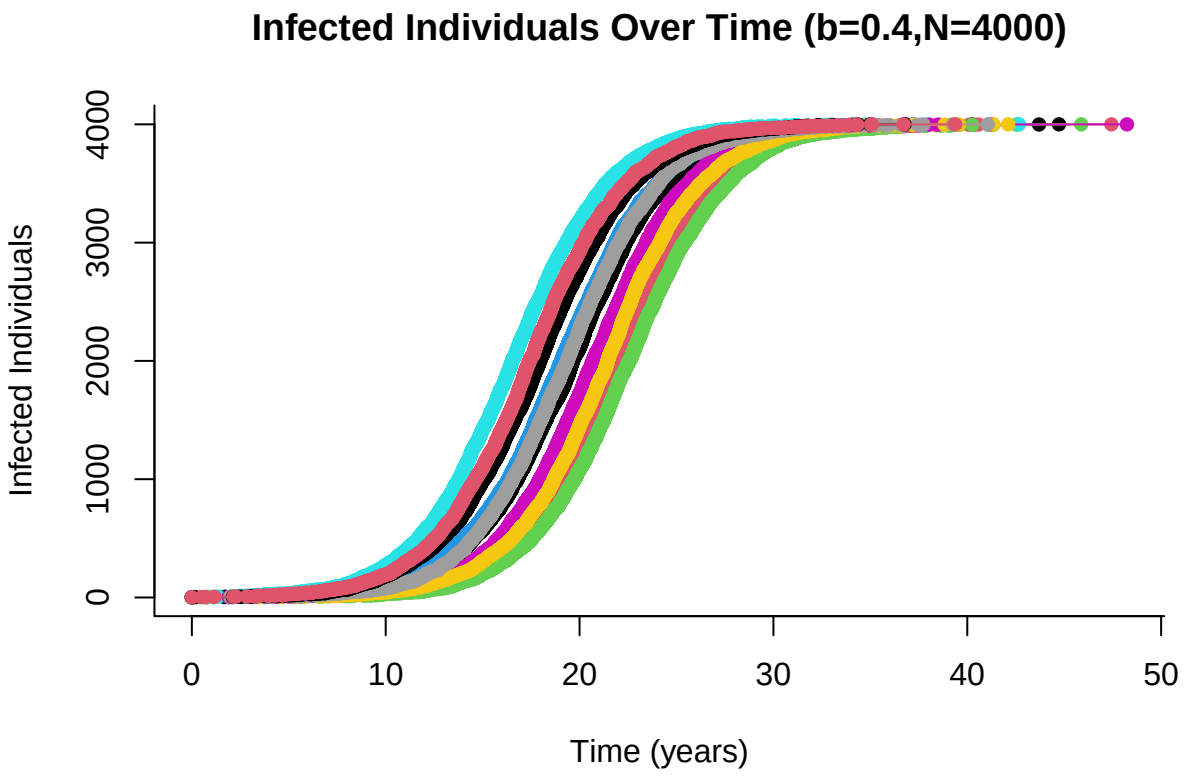
```r
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```




```r
# pop.size = 1000
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 1000  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.4, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.4,N=1000)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



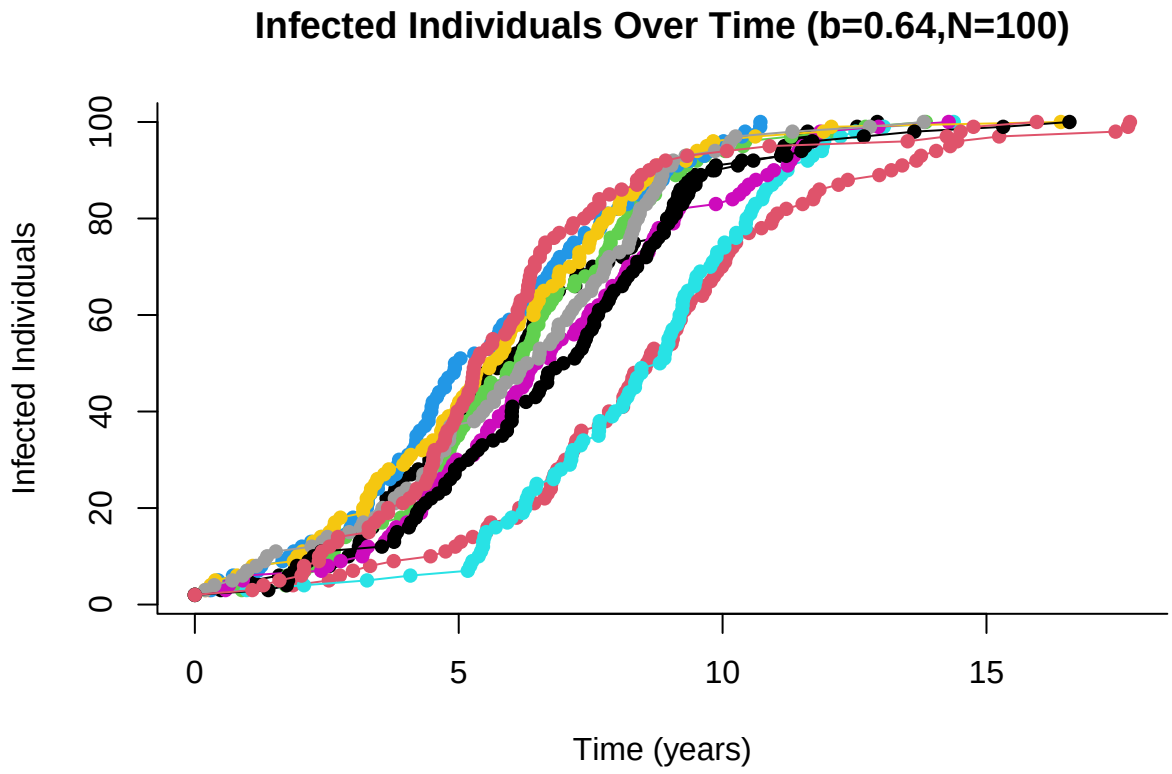

```r
# pop.size = 4000
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 4000  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.4, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.4,N=4000)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



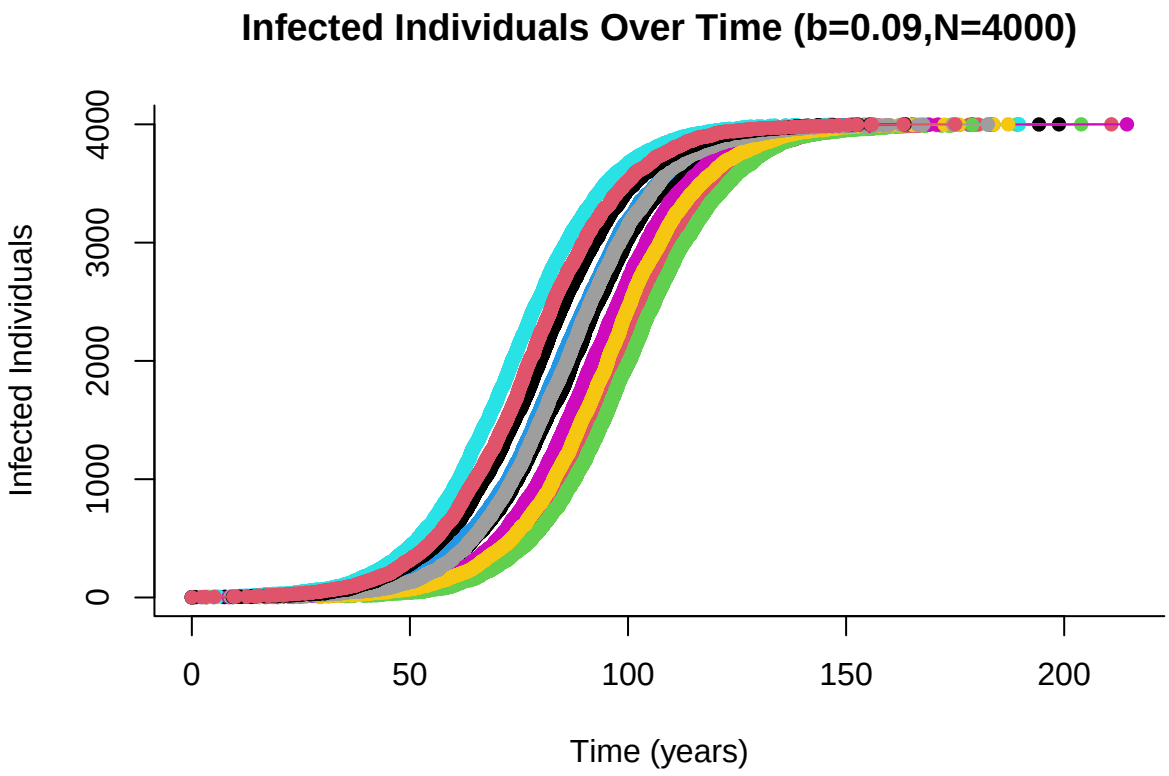
Lastly, I will show various models with converse values of beta and population size, for instance a large population with a small beta and vice versa. This showed a relatively drastic difference between the two scenarios as expected. When you do not have many individuals in your population but have a higher transmission parameter the population will take a shorter time to reach peak infection, such as in the case where some scenarios took just under 20 years to reach peak infection at N=100 and b=0.64. Conversely, with a low transmission parameter (0.09) and higher population (N=4000), the infection can take over 200 years in some scenarios to infect and entire population.

```
# beta = 0.64 pop.size = 100
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 100  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.64, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.64,N=100)")
```

```
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```




```
# beta = 0.09 pop.size = 4000
set.seed(38499583)  # make results repeatable
nsims <- 10  # number of simulations to run
pop.size <- 4000  # size of the population
Y0 <- 2  # initial number infected
nstep <- pop.size - Y0  # run until everyone infected
xstart <- c(time = 0, X = (pop.size - Y0), Y = Y0)  # initial conditions
params <- c(beta = 0.09, gamma = 365/13)  # parameters (R0=2.1)
simdat <- vector(mode = "list", length = nsims)  # to store simulated data
for (k in 1:nsims) {
    simdat[[k]] <- SI.simul(xstart, params, nstep)
}

trange <- range(sapply(simdat, function(x) range(x$time)))
yrange <- range(sapply(simdat, function(x) range(x$Y)))

plot(trange, yrange, type = "n", bty = "l", xlab = "Time (years)",
    ylab = "Infected Individuals", main = "Infected Individuals Over Time (b=0.09,N=4000)")
for (k in 1:nsims) lines(Y ~ time, data = simdat[[k]], col = k,
    type = "o", pch = 16)
```



Part 3

The deterministic models we previously studied were determined fully by the parameter values and the initial conditions we input. When compared to the stochastic models we have produced, one of the main differences is that along with these initial conditions and parameters, stochastic models will have an inherent randomness that can lead to a varying output while deterministic changes more steadily. These models have certain scenarios where they may be better suited, as stochastic is best in small populations and allows one to see possible individual events over time in those populations. While deterministic will produce a smoother model showing a more general trend overtime that may work best for larger populations.

Reference

Feldman, J., Mishra, S. What could re-infection tell us about R0? A modeling case-study of syphilis transmission. Infectious Disease Modelling. Volume 4, 2019, Pages 257-264. ISSN 2468-0427, https://doi.org/10.1016/j.idm.2019.09.002.

Extra Credit

Below I adapted the code from the given text (King, 2017) and adjust some parameters for changing events to ensure that the population remains fixed. I labeled them within the code where it begins to describe the type of events that may occur. So, if the event was a birth or death (of susceptible/infected/recovered) then there would be no change, as it previously was plus or minus one. When applying stochasticity to the SIR dynamics there are obvious ups and downs of model depending on random variation over time when

compared to a more deterministic approach or when just looking at the SI dynamics. The general curve shows that the amount of infected individuals will initially increase but begin to decrease as they recover over time. The stochastic nature for recovery can show possibly a more realistic model of not everyone recovering at a constant rate and may take a relatively longer or shorter time depending on the stochastic outcome.

```
SIR.onestep <- function (x, params) {
  X <- x[2] #susceptible
  Y <- x[3] #infected
  Z <- x[4] #recovered
  N <- X+Y+Z
  beta <- params["beta"]
  mu <- params["mu"]
  gamma <- params["gamma"]
  #each individual rate
  rates <- c(
    birth=mu*N,
    infection=beta*X*Y/N,
    recovery=gamma*Y,
    sdeath=mu*X,
    ideath=mu*Y,
    rdeath=mu*Z
  )
  #what changes with each event?
  transitions <- list(
    birth=c(0,0,0),      #CHANGED TO BE ZERO FOR NO BIRTHS
    infection=c(-1,1,0),
    recovery=c(0,-1,1),
    sdeath=c(0,0,0),     #CHANGED TO BE ZERO FOR NO DEATHS
    ideath=c(0,0,0),     #CHANGED TO BE ZERO FOR NO DEATHS
    rdeath=c(0,0,0)      #CHANGED TO BE ZERO FOR NO DEATHS
  )
  #total event rate
  total.rate <- sum(rates)
  #waiting time
  if (total.rate == 0)
    tau <- Inf
  else
    tau <- rexp(n=1,rate=total.rate)
  #which event occurs?
  event <- sample.int(n=6,size=1,prob=rates/total.rate)
  x+c(tau,transitions[[event]])
}
SIR.simul <- function (x, params, maxstep = 10000) {
  output <- array(dim=c(maxstep+1,4))
  colnames(output) <- names(x)
  output[1,] <- x
  k <- 1
  #loop until either k > maxstep or
  #there are no more infectives
  while ((k <= maxstep) && (x["Y"] > 0)) {
    k <- k+1
    output[k,] <- x <- SIR.onestep(x,params)
  }
  as.data.frame(output[1:k,])
```

```
}
set.seed(56856583)
nsims <- 10
xstart <- c(time=0,X=392,Y=8,Z=0) #initial conditions
params <- c(mu=0.02,beta=60,gamma=365/13) #parameters

require(plyr)
simdat <- rdply(
  nsims,
  SIR.simul(xstart,params)
)
head(simdat)
```

```
##   .n        time   X  Y Z
## 1  1 0.000000000 392  8 0
## 2  1 0.002728883 391  9 0
## 3  1 0.007003707 390 10 0
## 4  1 0.008856696 390  9 1
## 5  1 0.010046872 389 10 1
## 6  1 0.010658383 389  9 2
```

```
plot(Y~time,data=simdat,type='n')
d_ply(simdat,".n",function(x)lines(Y~time,data=x,col=.n))
```