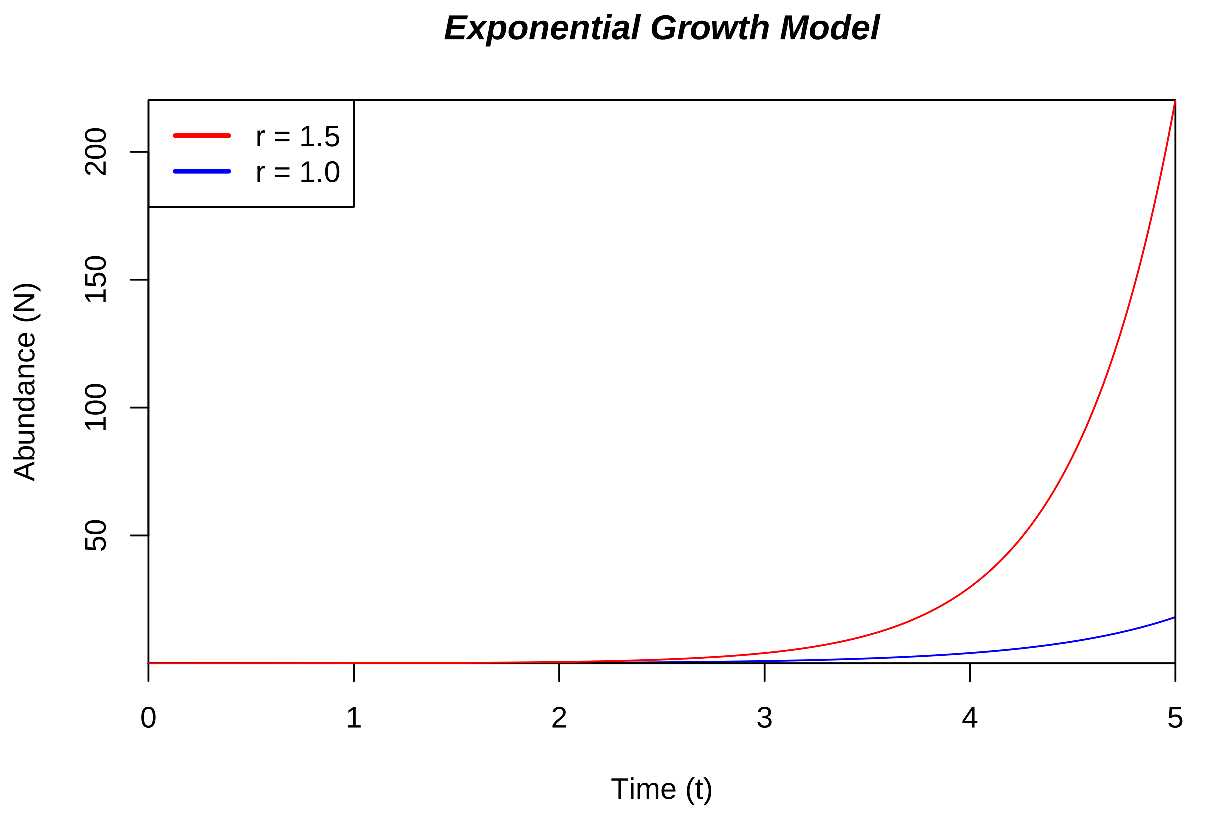
Lab 1

Clint Valentine

Numerically Solving Differential Equations

This block of code computes the exponential model $N(t) = N_0 e^{rt}$ using the R import `library(deSolve)`. Two conditions are plotted where $r = 1.5$ and $r = 1.0$ over the course of $\Delta time = 5$ with the initial condition $N_0 = 0.01$.

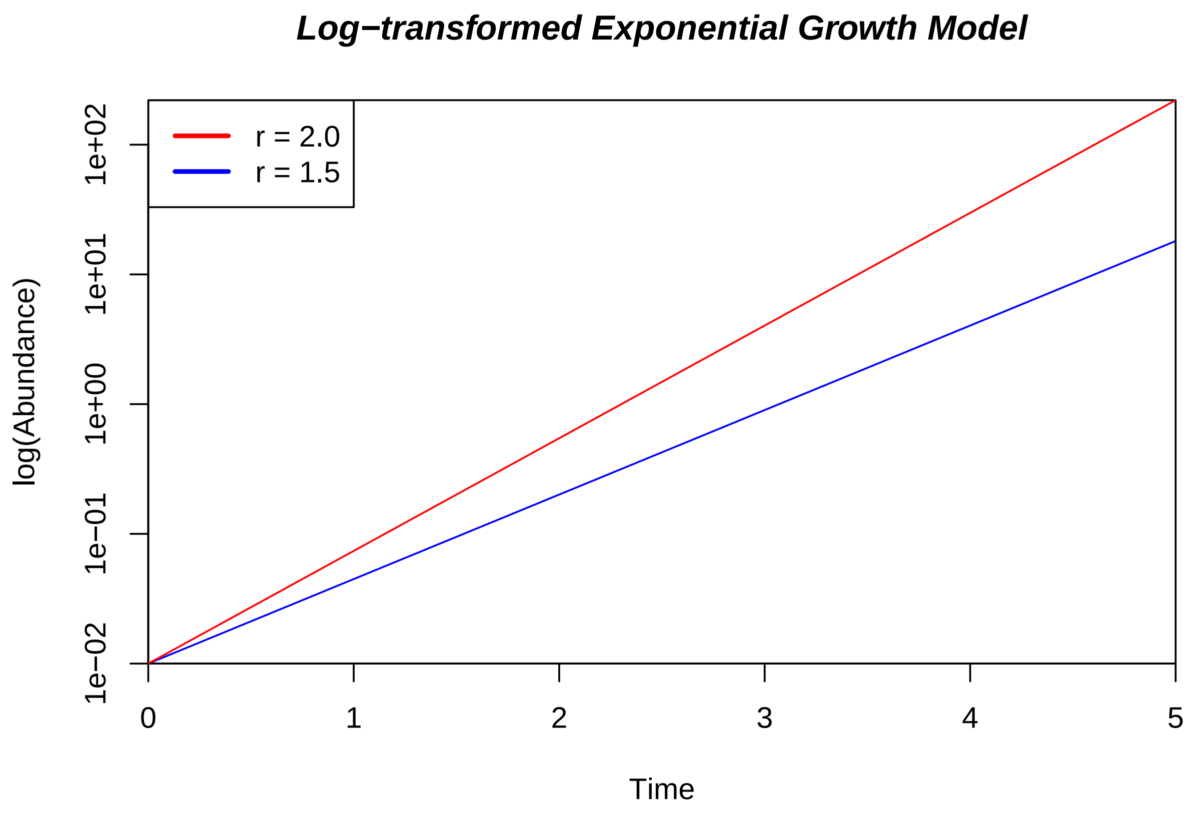
```
library(deSolve)

solve.expo <- function(t, y, parms) {
  dY <- parms$r * y
  return(list(dY))
}

No <- 0.01

total.time <- 5

parms.a <- list(r=1.5)
parms.b <- list(r=2.0)

results.a <- ode(y=No, times=seq(from=0, to=total.time, len=1000),
                 func=solve.expo, parms.a, method='ode45')

results.b <- ode(y=No, times=seq(from=0, to=total.time, len=1000),
                 func=solve.expo, parms.b, method='ode45')

plot(results.a[,1], results.a[,2], ylab='Abundance (N)', xlab='Time (t)',
     type='l', col='blue', ylim=range(results.a[,2], results.b[,2]),
     yaxs='i', xaxs='i')

lines(results.b[,1], results.b[,2], col='red')

title(main ='Exponential Growth Model', col.main='black', font.main=4)

legend(x='topleft', legend=c('r = 1.5', 'r = 1.0'), lty=c(1,1), lwd=c(2.5,2.5),
       col=c("red","blue"))
```



This figure supports our understanding of the exponential model because as the intrinsic capacity for increase r in e^{rt} increases the exponential rate of abundance N should increase linearly.

We can determine if our hypothesis is validated by plotting a linear regression through the log-transformed data for each simulation. The slope of each regression should be equal to our values of r. The y-intercept of both equations should also be equal to $N_0 = 0.01$.

```r
plot(results.a[,1], results.a[,2], log='y', ylab='log(Abundance)', xlab='Time',
     type='l', col='blue', ylim=range(results.a[,2], results.b[,2]),
     yaxs='i', xaxs='i')

lines(results.b[,1], results.b[,2], col='red')

title(main ='Log-transformed Exponential Growth Model', col.main='black',
      font.main=4)

legend(x='topleft', legend=c('r = 2.0', 'r = 1.5'),
       lty=c(1,1), lwd=c(2.5,2.5), col=c('red', 'blue'))
```



```
# Linear regressions to the log transformed simulation data
lm.a <- lm(log(results.a[,2]) ~ results.a[,1])
lm.b <- lm(log(results.b[,2]) ~ results.b[,1])

# The log-normalized y-intercept of both linear regressions
# fit to the log transformed data
print(exp(unname(coef(lm.a)[1]))) # Should be No = 0.01
```

```
## [1] 0.01
```

```
print(exp(unname(coef(lm.b)[1]))) # Should be No = 0.01
```

```
## [1] 0.01
```

```
# The slope of both linear regressions fit to the log transformed data
print(unname(coef(lm.a)[2]))      # Should be r = 1.5
```

```
## [1] 1.5
```

```
print(unname(coef(lm.b)[2]))      # Should be r = 2.0
```

```
## [1] 2
```

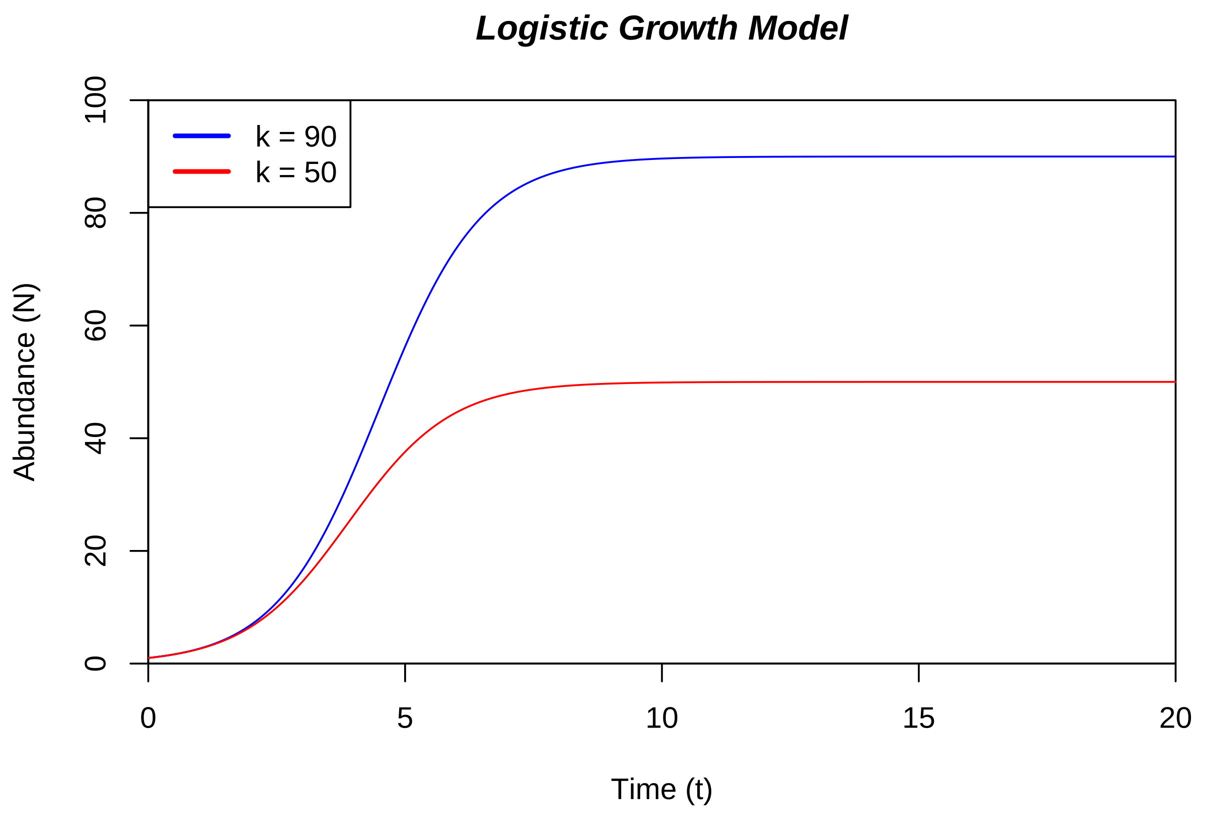
The function below simluates the dynamics of the logistic model $\frac{dN}{dt} = rN_o(1 - \frac{N}{K})$

```
# Function to simulate logistic growth
solve.logistic <- function(t, y, parms) {
  dY <- parms$r * y * (1 - (y / parms$k))
  return(list(dY))
}
```

The below manipulation function was set to not execute `{r eval=FALSE}`. This function was used to find values of two simulations that show how temporal dynamics are affected by varying only the carrying capacity of the logistic growth model.

```
library(manipulate)
manipulate ({
  # Code that needs to be manipulated
  parms <- list(r=rVal, k=kVal)
  times <- 0:tVal
  results <- ode(y=NVal, times=times, func=solve.logistic, parms,
                 method="ode45")

  plot(results[,1], results[, 2], xlab="Time (t)", ylab="Abundance (N)",
       type="l", col="blue")
},
  # The graphical user interface elements used to change the parameter values
  tVal=slider(min=20, max=500, initial=20, label="Total timesteps (t)"),
  rVal=slider(min=0, max=4, initial=0.35, step=0.05, label="Growth rate (r)"),
  NVal=slider(min=0, max=20, initial=1, label="Initial abundance (N0)"),
  kVal=slider(min=0, max=100, initial=100, label="Carrying Capacity (K)"))
```

```
No <- 1

total.time <- 20

parms.a <- list(r=1, k=90)
parms.b <- list(r=1, k=50)

results.a <- ode(y=No, times=seq(from=0, to=total.time, len=1000),
                 func=solve.logistic, parms.a, method='ode45')

results.b <- ode(y=No, times=seq(from=0, to=total.time, len=1000),
                 func=solve.logistic, parms.b, method='ode45')

plot(results.a[,1], results.a[,2], ylab='Abundance (N)', xlab='Time (t)',
     type='l', col='blue', ylim=c(0, 100), yaxs='i', xaxs='i')

lines(results.b[,1], results.b[,2], col='red')

title(main ='Logistic Growth Model', col.main='black', font.main=4)

legend(x='topleft', legend=c('k = 90', 'k = 50'),
       lty=c(1,1), lwd=c(2.5,2.5), col=c('blue', 'red'))
```

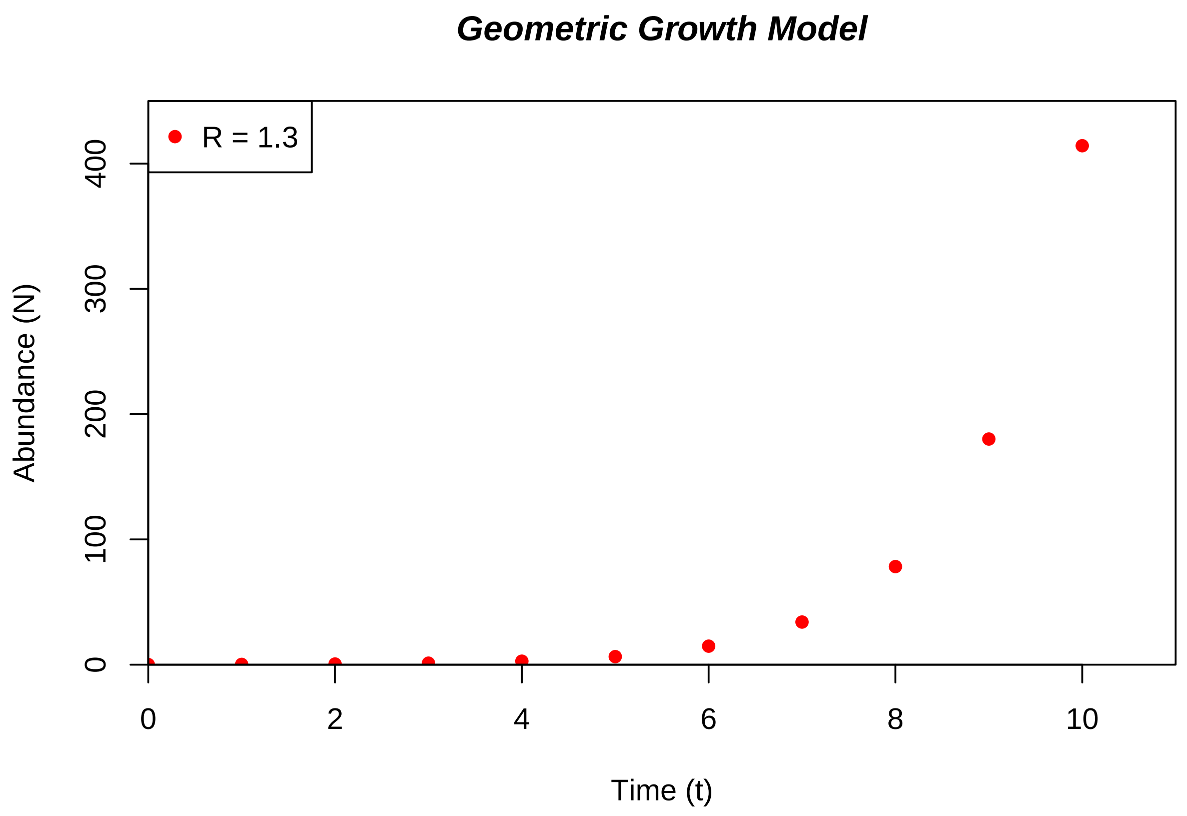


The figure above was simulated using the logistic growth function with the parameters: $N_o = 1$, $r = 1$, and $\Delta time = 20$. It appears as if the specific carrying capacity values are independant of when a logistic growth system reaches it's set carrying capacity. The rate of increase is slower in systems with a low carrying capacity while the intrinisic capacity for increase remains constant.

Simulating the Dynamics of Discrete-Time Models

Geometric growth can be simulated using a `for` loop as shown in the following example. In this case $R = 1.3$ or $\lambda = 2.3$ over the course of $\Delta time = 10$ with the initial condition $N_0 = 0.1$. The time resolution for this model was set to 1 so the geometric growth is visible within ten times steps.

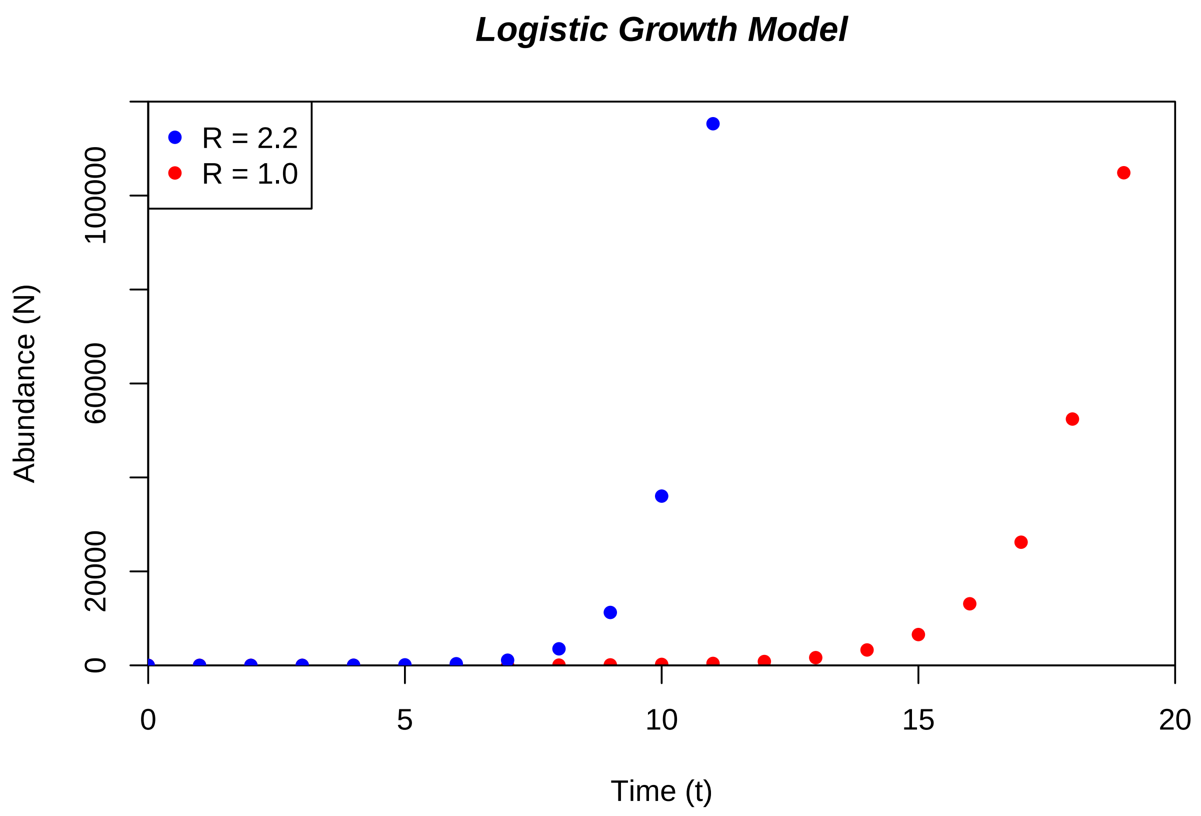
```
time.res = 1

time <- seq(0, 10, by=time.res)

N <- 0.1

parms <- list(R=1.3)

abundance <- numeric(length(time))

abundance[1] <- N

for (x in seq(length(time) - 1)) {
  abundance[x + 1] <- abundance[x] * (1 + parms$R)
}

plot(time, abundance, ylab='Abundance (N)', xlab='Time (t)',
     pch=16, col='red', ylim=c(0, 450), xlim=c(0, 11), yaxs='i', xaxs='i')

title(main ='Geometric Growth Model', col.main='black',
      font.main=4)

legend(x='topleft', legend='R = 1.3',
       pch=16, col='red')
```



This block of code simulates the logistic growth equation $N(t) = N_0\lambda^t$ using `for loops`. In the following example $r = 1.0$ or $\lambda = 2.0$ over the course of $\Delta time = 20$ with the initial condition $N_0 = 0.1$. Once again, rhe time resolution for this model was set to 1 so the geometric growth is visible within the twenty time steps.

```
time.res = 1

time <- seq(0, 20, by=time.res)

N <- 0.1

parms.a <- list(R=1.0)
parms.b <- list(R=2.2)

abundance.a <- numeric(length(time))
```

```r
abundance.b <- numeric(length(time))

for (x in seq(length(time))) {
  abundance.a[x] <- N * (parms.a$R + 1) ^ x
  abundance.b[x] <- N * (parms.b$R + 1) ^ x
}

plot(time, abundance.a, ylab='Abundance (N)', xlab='Time (t)',
     pch=16, col='red', ylim=c(0,120000), yaxs='i', xaxs='i')

points(time, abundance.b, col='blue', pch=16)

title(main ='Logistic Growth Model', col.main='black',
      font.main=4)

legend(x='topleft', legend=c('R = 2.2', 'R = 1.0'),
       pch=16, col=c('blue', 'red'))
```



The dynamics of the above figure support our understanding of the logistic growth function. The finite rate of increase $\lambda = R + 1$ is directly proportional to the rate of geometric growth. This value can be seen as a multiplyer that accelerates the rate of increase when $\lambda > 1$ and deccelerates the rate of increase when $\lambda < 1$.

The function below simluates the dynamics of the logistic map model in discrete time $N_{t+1} = N_t(1+R(1-\frac{N_t}{K}))$

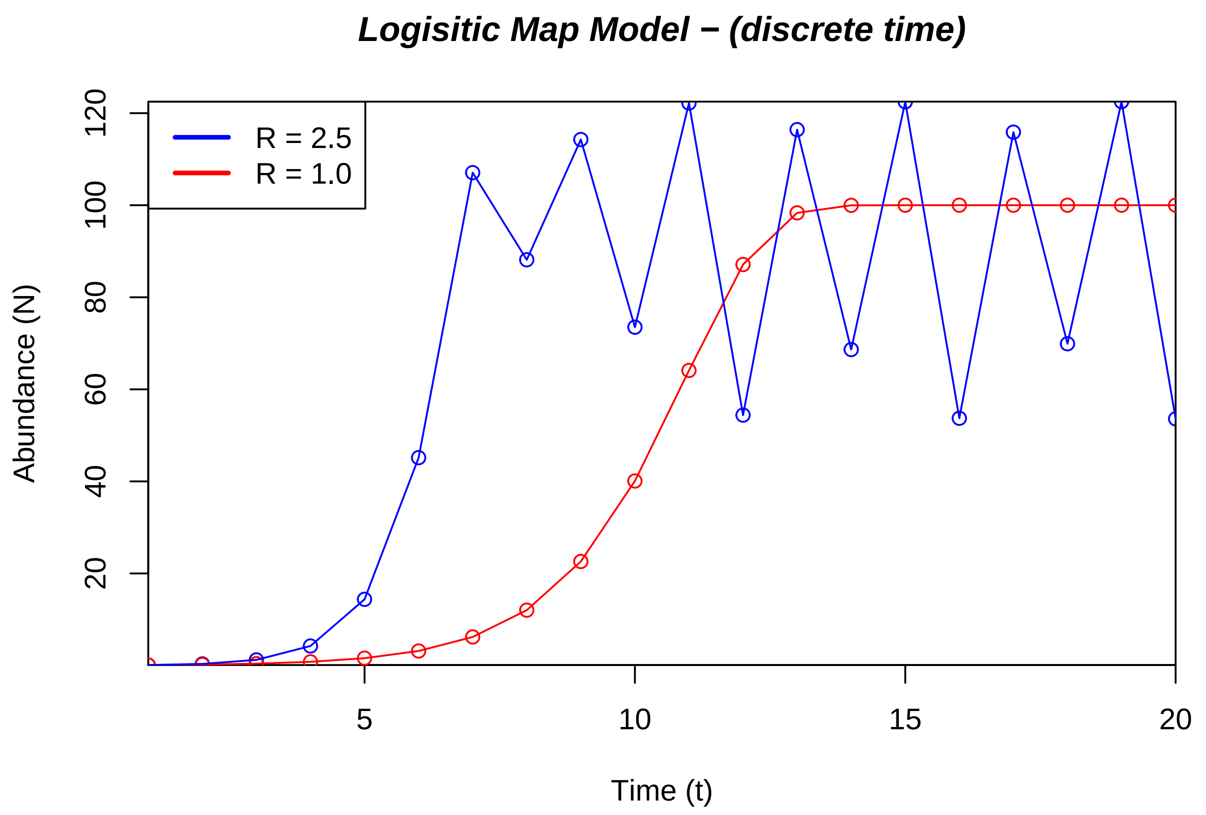
```r
# Function to simulate the logistic map
solve.logmap <- function(Nt, R, K) {
  N <- Nt * (1 + R * (1 - Nt/K))
  return(N)
}
```

The below manipulation function was set to not execute `{r eval=FALSE}`. The following example is used to interactively modify the value of R in the simulated logistic map from the values $1 : 3.3$.

```r
library(manipulate)
manipulate ({
  # Code that needs to be manipulated
  parms <- list(R=rVal)
  time <- 0:tVal
  K <- kVal
  No <- NVal

  abundance <- numeric(length(time))
  abundance[1] <- No

  for (x in 2:length(time)) {
    abundance[x] <- solve.logmap(abundance[x - 1], parms$R, K)
  }

  plot(time, abundance, ylab='Abundance (N)', xlab='Time (t)',
     type='l', col='red', yaxs='i', xaxs='i')
  points(time, abundance, pch=1, col='red')
},
  # The graphical user interface elements used to change the parameter values
  tVal=slider(min=1, max=20, initial=20, label="Total timesteps (t)"),
  rVal=slider(min=0, max=3.3, initial=1.5, step=0.1, label="Finite rate of increase (R)"),
  NVal=slider(min=10, max=200, initial=30, step=10, label="Initial abundance (N0)"),
  kVal=slider(min=10, max=200, initial=50, step=10, label="Carrying Capacity (K)"))
```

```r
time <- 1:20

No <- 0.1
K <- 100

parms.a <- list(R=1.0)
parms.b <- list(R=2.5)

abundance.a <- numeric(length(time))
abundance.b <- numeric(length(time))

abundance.a[1] = No
abundance.b[1] = No

for (x in 2:length(time)) {
  abundance.a[x] <- solve.logmap(abundance.a[x - 1], parms.a$R, K)
  abundance.b[x] <- solve.logmap(abundance.b[x - 1], parms.b$R, K)
}
```

```r
plot(time, abundance.b, ylab='Abundance (N)', xlab='Time (t)',
      pch=1, col='blue', yaxs='i', xaxs='i')

points(time, abundance.a, pch=1, col='red')

lines(time, abundance.a, col='red')
lines(time, abundance.b, col='blue')

title(main ='Logisitic Map Model - (discrete time)', col.main='black', font.main=4)

legend(x='topleft', legend=c('R = 2.5', 'R = 1.0'),
       lty=c(1,1), lwd=c(2.5,2.5), col=c('blue', 'red'))
```

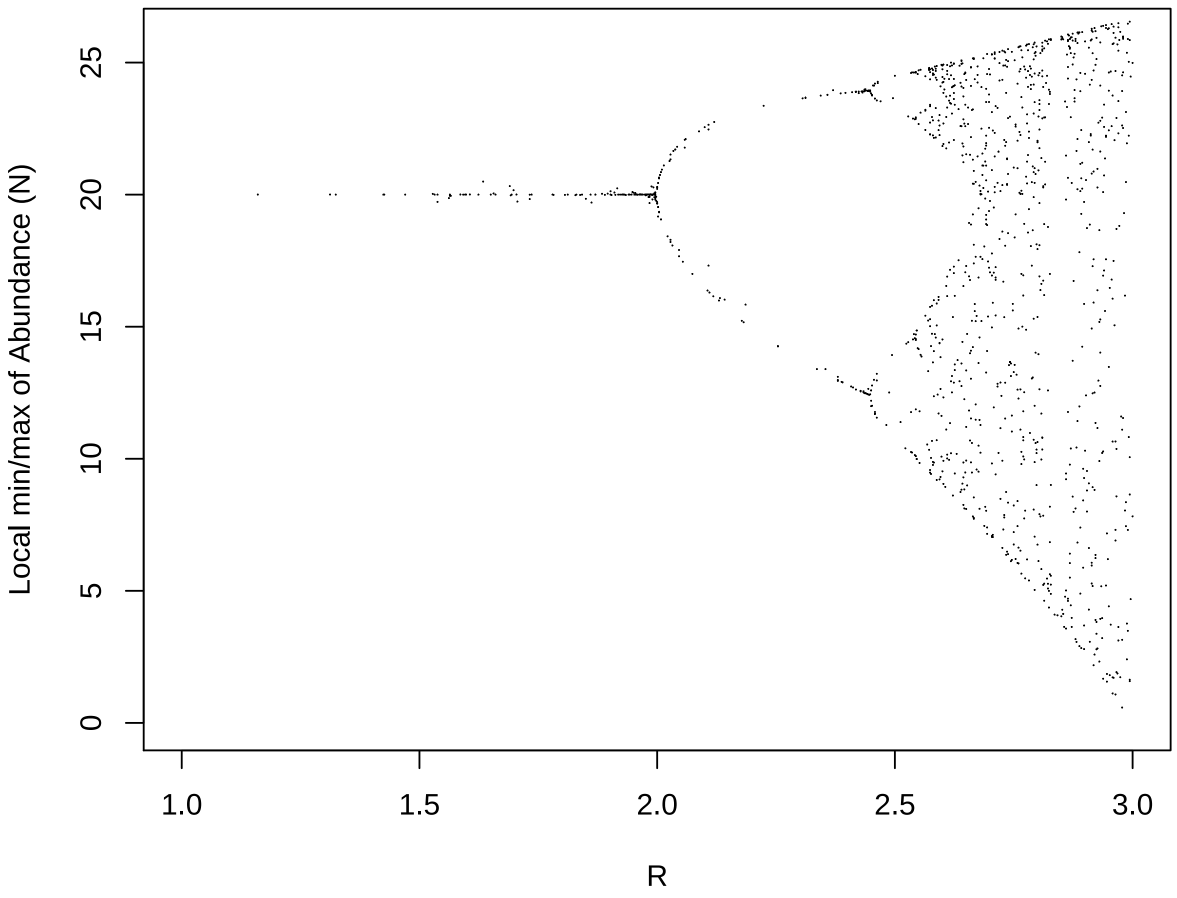


Two simulations were run above using the same logistic map model with a difference in R values. The red line with $R = 1.0$ shows a more relaxed form of the logistic map model compared to the increased growth of the model with $R = 2.5$ and the eventual chaotic behavior about the carrying capacity K. The reason for this difference is in the form of the discrete-time equation behind the model as described above.

The following code chunk simulates every 500th of R for $1.0 < R < 3.0$. Each simulation was set with a carrying capacity of twenty and an initial abundance at 0.5 and spanned 1000 discrete time steps.

For each simulation, the local maximum, minimum, and equilibrium points are found and stored. The resulting plot shows a bifurcation diagram for each R value and the cycling points found for each R value. We will call the `solve.logmap` function defined earlier.

```r
time <- 1:1000

No <- 0.5
K <- 20

rates <- seq(1, 3, by=(3 - 1) / 1000)

solve.forR <- function(R) {
  # Setup abundance vector for passed R
  a <- numeric(length(time))
  a[1] = No

  # Solve per the logistic map
  for (x in 2:length(time)) {
    a[x] <- solve.logmap(a[x - 1], R, K)
  }

  # Initialize values to be stored per loop
  values <- numeric()

  # Scan through time and check to see if a point is a local
  # maximum, minimum, or equilibrium point
  for (x in 2:(length(a) - 1)) {
    locmin = (a[x] < a[x - 1]) & (a[x] < a[x + 1])
    locmax = (a[x] > a[x - 1]) & (a[x] > a[x + 1])
    stability = (a[x] == a[x - 1]) & (a[x] == a[x + 1])

    # Append to found values
    if (locmin | locmax | stability) {
      values <- c(values, a[x])
    }
  }
  return(unique(values))
}

# Open empty plot to predetermined size
plot(NULL, xlim=c(1,3), ylim=c(0,26), ylab="Local min/max of Abundance (N)",
     xlab="R")

# Main loop that finds the max, min, and equil. values of (N) for each R
for (R in rates) {
  Y <- solve.forR(R)
  X <- rep(R, length.out=length(Y))
  points(X, Y, col='black', cex=0.00001)
}
```


Local min/max of Abundance (N)
0
5
10
15
20
25
1.0
1.5
2.0
2.5
3.0
R