# Lab 02


### Jennifer Lin

### jenniferyjlin@berkeley.edu (mailto:jenniferyjlin@berkeley.edu)

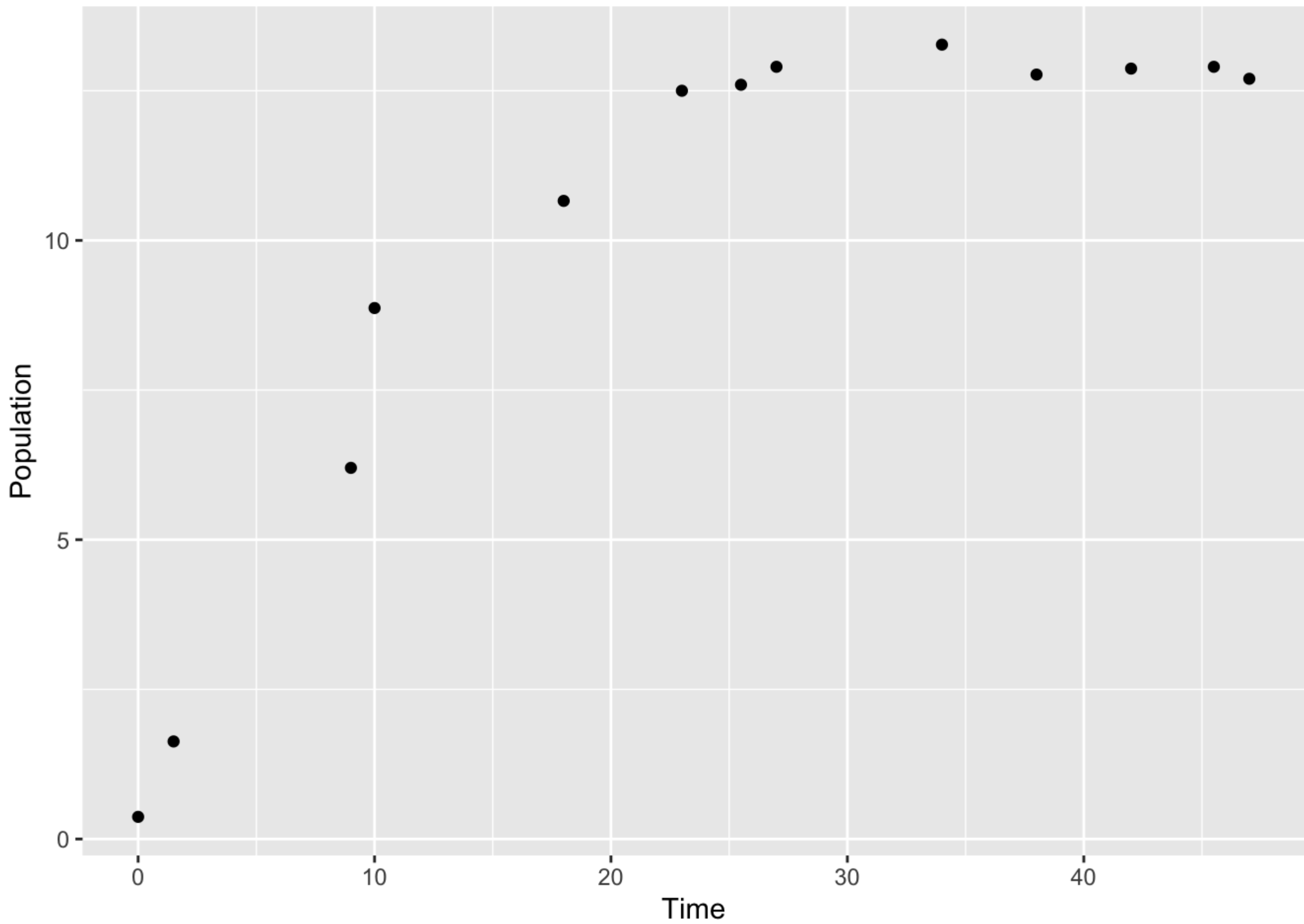

### Import data

```
yeast_data = read.table("lab_02_data.txt", header=T)
```

# 1

### Plot the yeast data given in the table.

```
library(ggplot2)
q1 <- ggplot(data=as.data.frame(yeast_data),aes(Time,Population)) + geom_point()
print(q1)
```




# 2

### What is the approximate carrying capacity of yeast?

```
# According to the plot, the approximate carrying capacity K of yeast is 13 unit of t
he population.
```

# 3

## 3. Through trial and error, fit a logistic model of growth to the data presented in the table.

### To do this use the carrying capacity you approximated from part 1b,

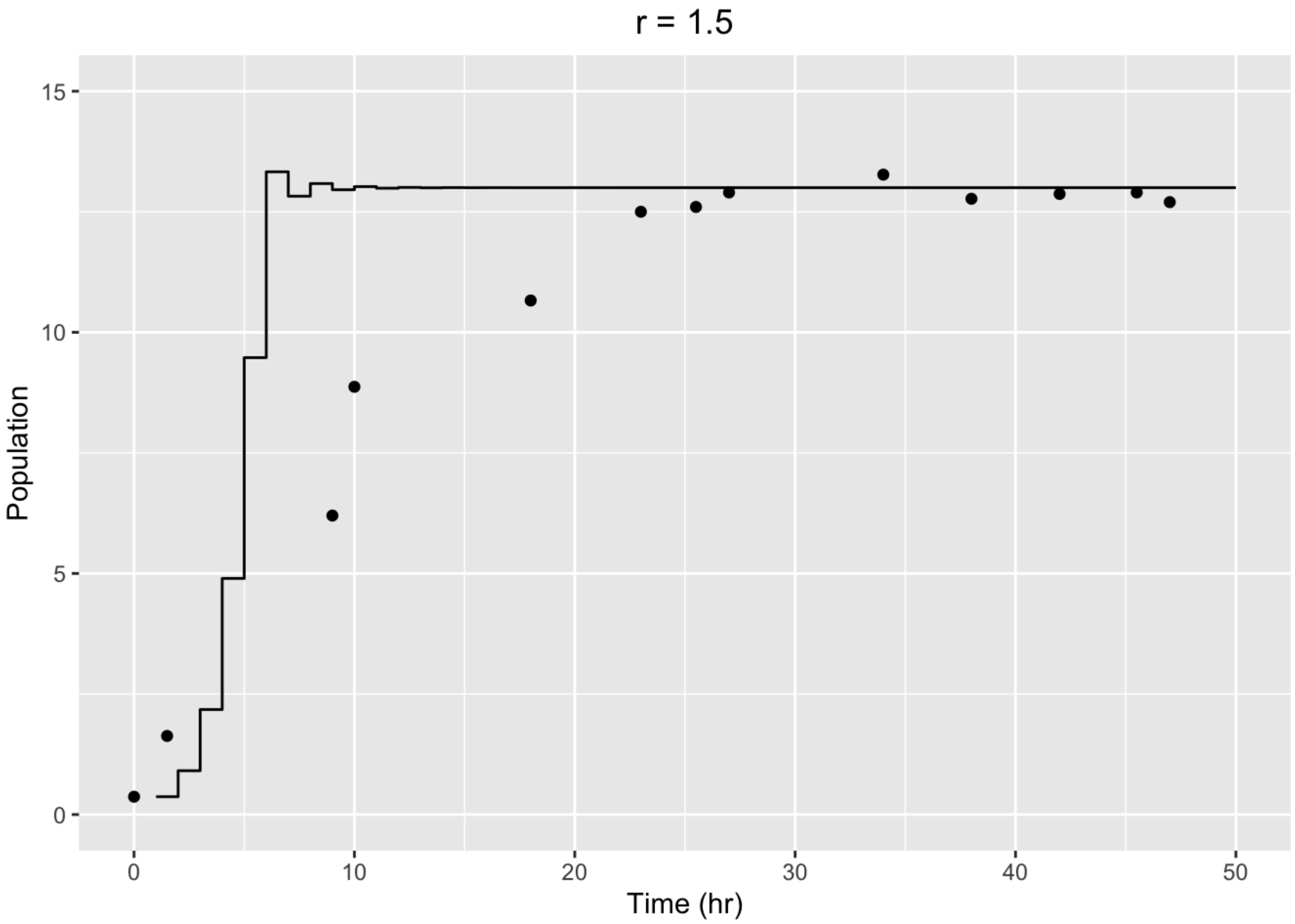
### the initial condition from the data and

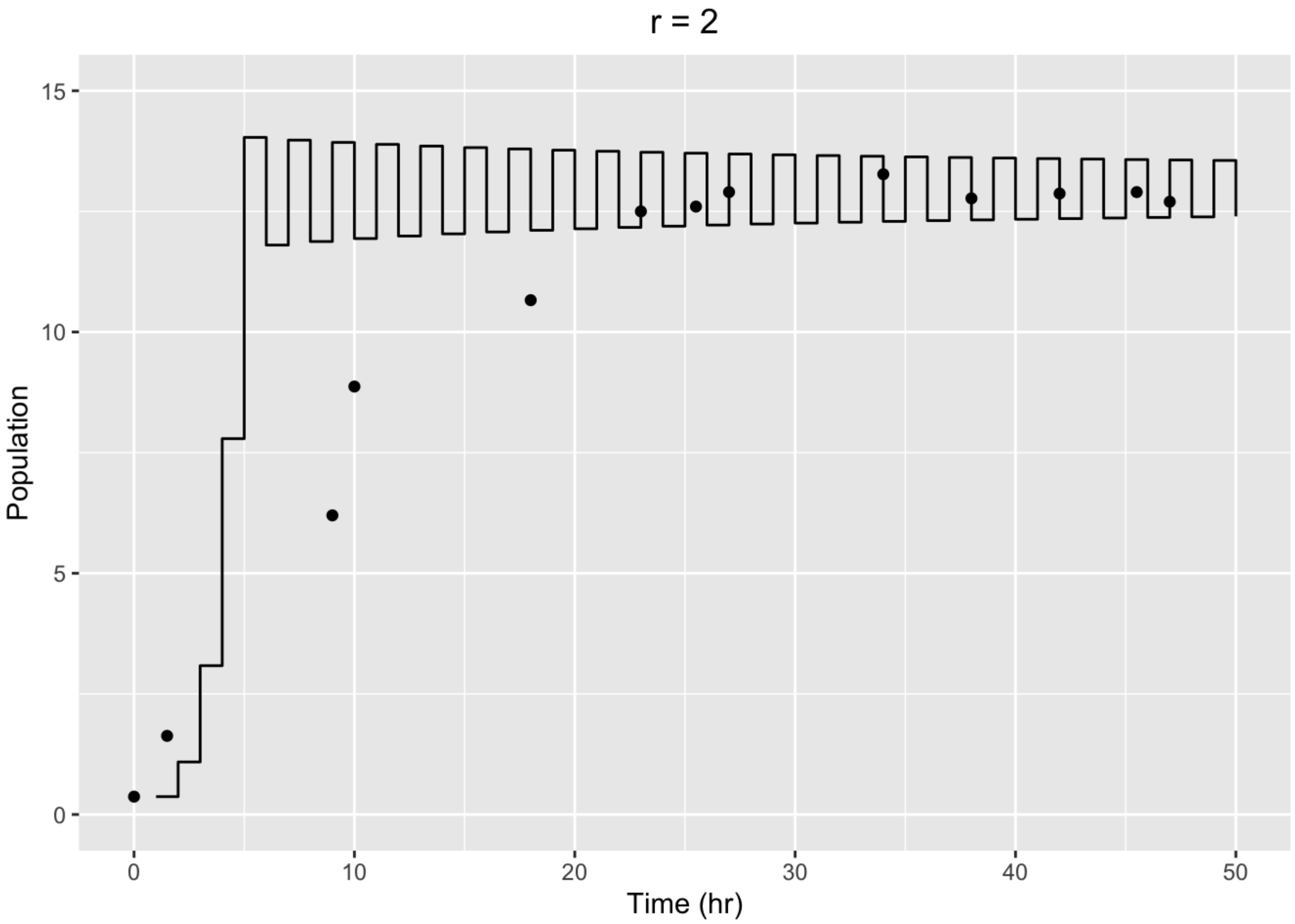
### try the following reproductive factors (r = 1.5, r = 2.0, r = 0.4, r = 0.7).

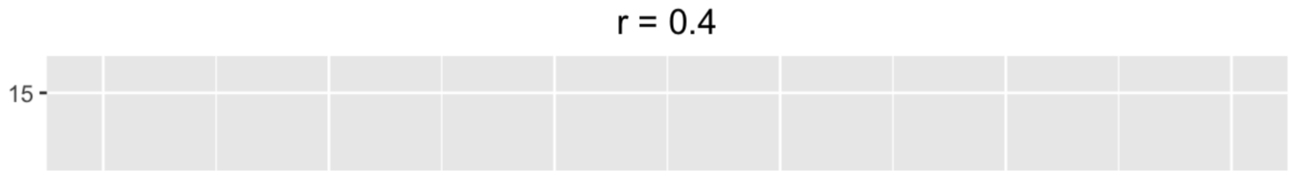
### Which reproductive factor fits the data best?

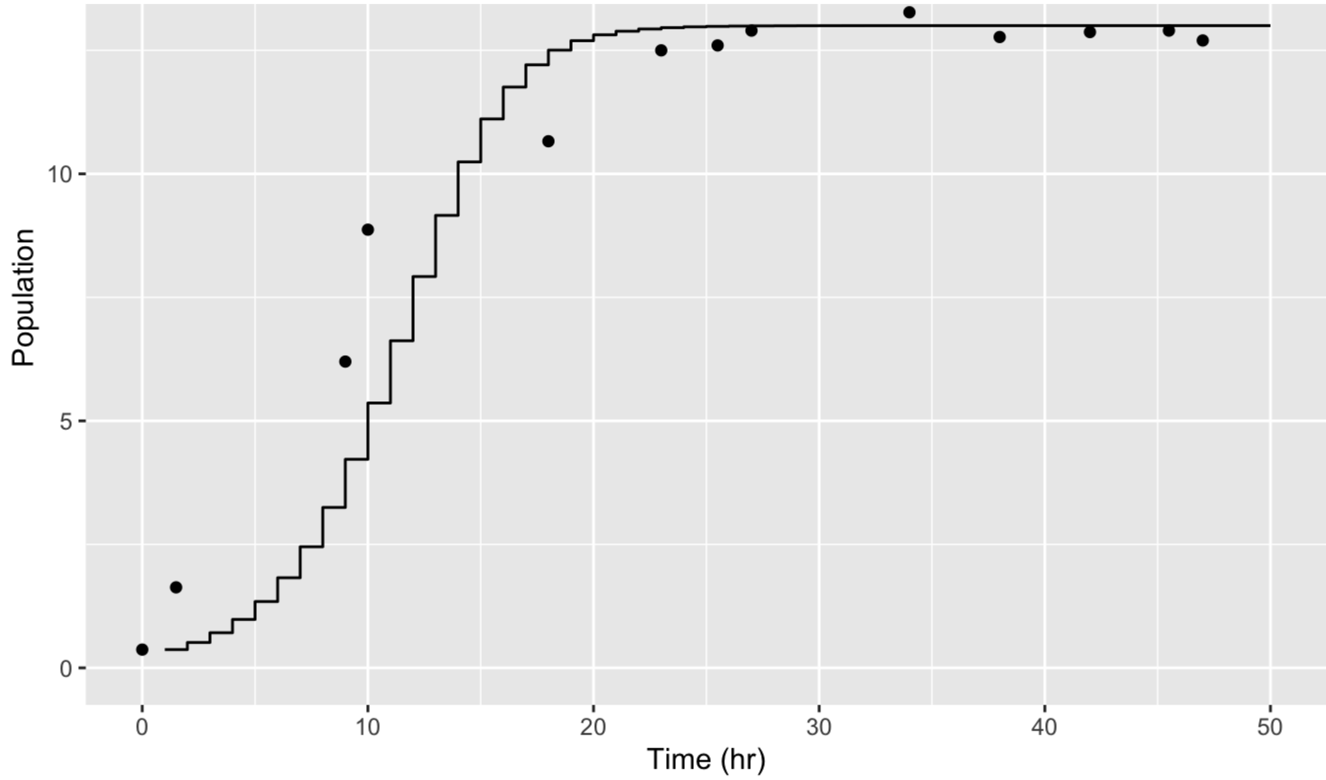
### (Hint: use R to iteratively solve the recursive logistic equation. This can be done with a single loop).

```r
rList <- c(1.5, 2.0, 0.4, 0.7)
K <- 13
for (j in 1:4){
  r <- rList[j]
  current_n <- c()
  current_n[1] <- 0.37
  last_n <- current_n[1]
  for (i in 2:50){
    current_n[i] = last_n + r*last_n*(1-last_n/K)
    last_n <- current_n[i]
  }
  current_n_df <- cbind(1:50, current_n)
  colnames(current_n_df) <- c("Time","Population")
  current_n_df <- data.frame(current_n_df)
  # plot
  plot <- ggplot(NULL, aes(Time, Population)) +
    geom_point(data = yeast_data) +
    geom_step(data = current_n_df) +
    ggtitle(paste0("r = ",rList[j])) +
    theme(plot.title = element_text(hjust = 0.5)) +
    xlab("Time (hr)") +
    ylim(0, 15)
  print(plot)
  assign(paste0("q3_r",rList[j]),plot)
}
```

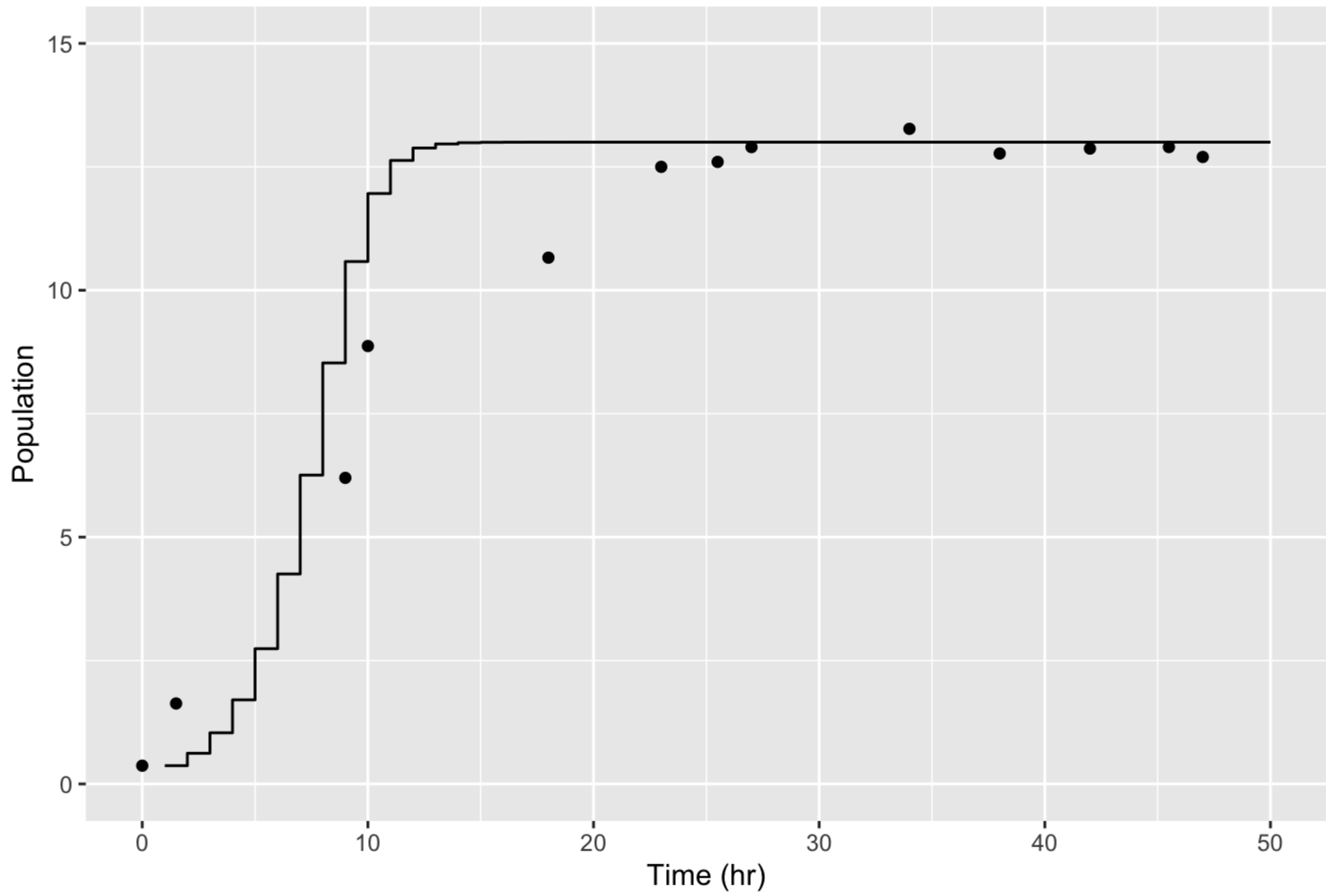

r = 1.5
15
10
5
0
Population
0
10
20
30
40
50
Time (hr)


r = 2
15
10
5
0
Population
0
10
20
30
40
50
Time (hr)


r = 0.4
15




r = 0.7




```
# r = 0.4 fits the data best
```

# 4

Suppose you have decided to go into the yeast selling business.

Modify the logistic growth equation to include a constant daily harvest amount, denote this amount h.

Bcouese note: add a subtraction term to the recursion (so original expression - h) in every iteration of the loop and you are not obligated to use a conditional statement like I taught in class to account for every 24 hours.

```
# Daily harvest amount = h
# current_n = last_n + r*last_n*(1-last_n/K) - h
```

# 5

Mathematically find the new equilibrium points for this new model.

```
# (Hint: You will likely have to use the quadratic equation, x = (-b ± (b^2-4*a*c)^
(1/2) ) / 2*a, to find the equilibrium points).
# equilibrium points -> current_n = last_n
# current_n = last_n = last_n + r*last_n*(1-last_n/K) - h
# last_n = last_n + r*last_n*(1-last_n/K) - h
# let last_n = x
# x = x + r*x*(1-x/K) - h
# 0 = r*x*(1-x/K) - h (-x on each side)
# 0 = r*x*(1) - r*x*(x/K) - h
# r*x*x/K - r*x + h = 0
# r*x*x - r*K*x + h*K = 0 (*K on each side)
# a = r
# b = -r*K
# c = h*K
# x = (r*K + ((-r*K)^2-4*r*h*K)^(1/2) ) / (2*r)   OR    (r*K - ((-r*K)^2-4*r*h*K)^(1/
2) ) / (2*r)
```

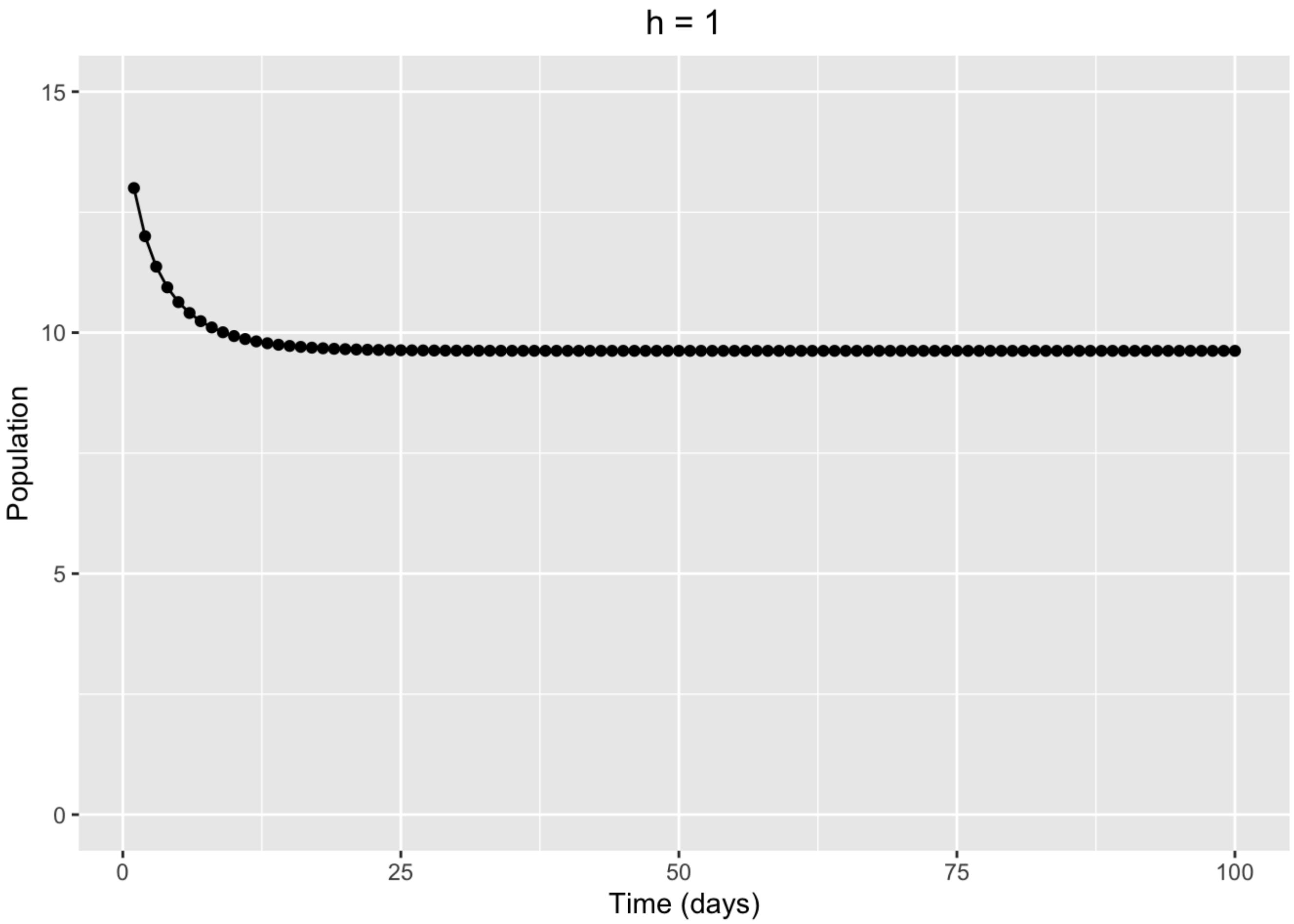
# 6

Set h = 1. Starting at the carrying capacity you found in part 1b.

(i.e. set the initial population size to K) simulate 100 days of yeast growth.

Is this a sustainable harvest amount? Make sure to plot your results to help visualize what is happening.

```r
K <- 13
r <- 0.4
h <- 1
current_n <- c()
current_n[1] <- K
last_n <- current_n[1]
for (i in 2:100){
  current_n[i] = last_n + r*last_n*(1-last_n/K) - h
  last_n <- current_n[i]
}
current_n_df <- cbind(1:100, current_n)
colnames(current_n_df) <- c("Time","Population")
current_n_df <- data.frame(current_n_df)
# plot
q6 <- ggplot(data=as.data.frame(current_n_df),aes(Time,Population)) +
  geom_line() +
  geom_point() +
  ggtitle(paste0("h = ",h)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Time (days)") +
  ylim(0, 15)
print(q6)
```




```
# h = 1 is a sustainable harvest amount. The population of yeast drops from 13 to ~9.
62 during 0-10 days and stays stable.
```

# 7

If you found h = 1 to be unsustainable in the long term, change the harvest amount so that it becomes sustainable.

Plug your values for h, r and K into the equilibrium points you found in part 2b. and see if your answer matches the simulation result.

```
# h = 1 is sustainable in the long term. Thus, we don't need to change the harvest am
ount.
K <- 13
r <- 0.4
h <- 1
x1 <- (r*K + ((-r*K)^2-4*r*h*K)^(1/2) ) / (2*r)
x2 <- (r*K - ((-r*K)^2-4*r*h*K)^(1/2) ) / (2*r)
print(paste0("The equilibrium points can be either ", x1, " or ", x2, ". Yet, both of
them are not equal to the results I simulated, which is ", last_n, "."))
```

```
## [1] "The equilibrium points can be either 9.6224989991992 or 3.3775010008008. Yet,
both of them are not equal to the results I simulated, which is 9.62249900051759."
```

```
print(paste0("Since I started the simulation at the carrying capacity K = ", K, ", th
e final equilibrium population size should be ", x1, ". This value is very close to t
he final state of my simulation ", current_n_df[100,2], "."))
```

```
## [1] "Since I started the simulation at the carrying capacity K = 13, the final equ
ilibrium population size should be 9.6224989991992. This value is very close to the f
inal state of my simulation 9.62249900051759."
```

# 8 BONUS

### Maximal periodic harvesting.

### Using simulations find the best harvesting strategy given periodically harvesting of some amount of yeast.

### How often should you harvest and what volume of yeast should you harvest?

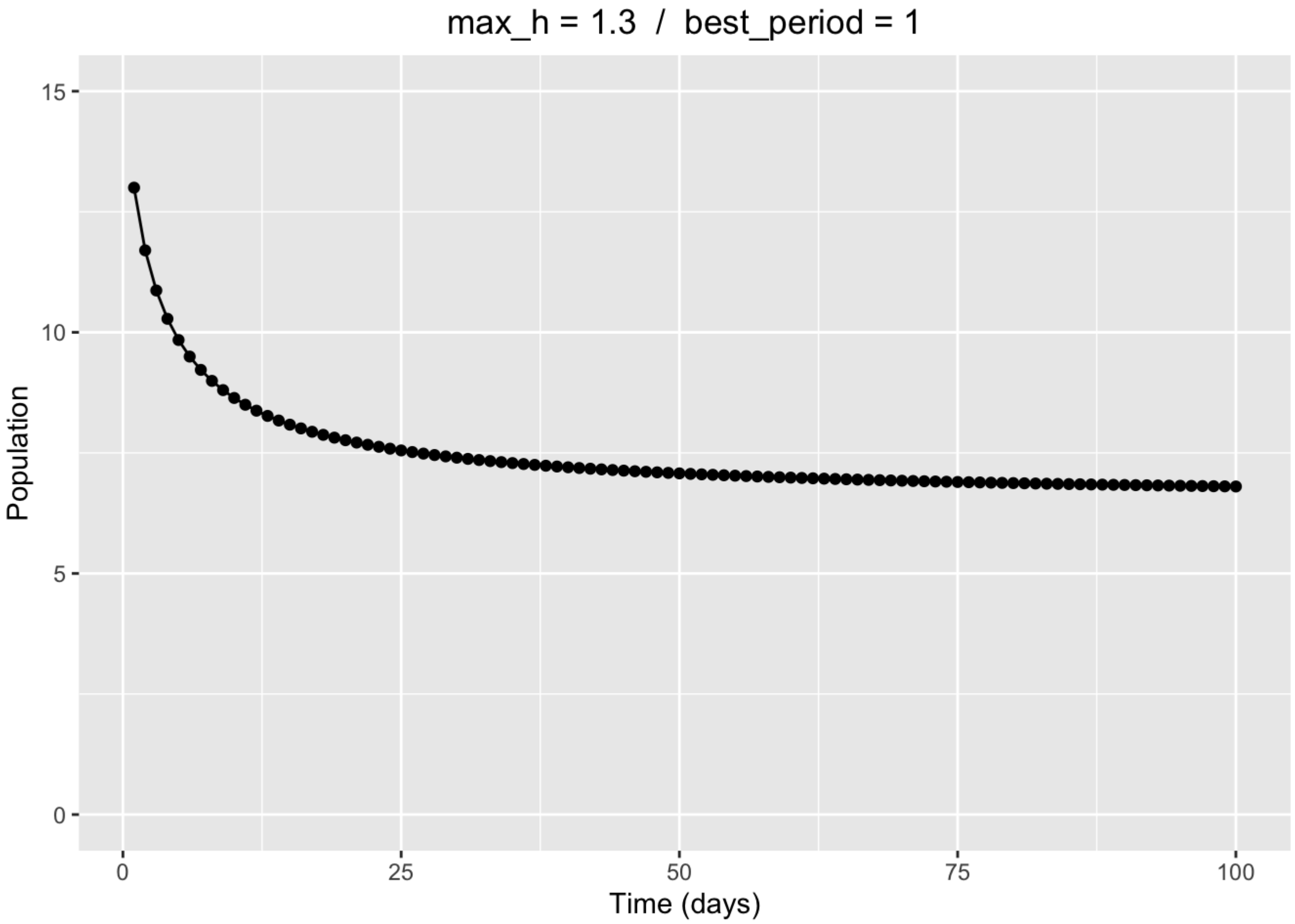
```
K <- 13
r <- 0.4
for(p in seq(10,1)){
  for(h in seq(0.5,3,0.1)){
    current_n <- c()
    current_n[1] <- K
    last_n <- current_n[1]
    for (i in 2:100){
      if (i%%p==0){
        current_n[i] = last_n + r*last_n*(1-last_n/K) - h
        last_n <- current_n[i]
      }
      else{
        current_n[i] = last_n + r*last_n*(1-last_n/K)
        last_n <- current_n[i]
      }
    }
    current_n_df <- cbind(1:100, current_n)
    colnames(current_n_df) <- c("Time","Population")
    current_n_df <- data.frame(current_n_df)
    if (any(is.infinite(current_n_df[,2]))){}
    else if (sd(current_n_df[80:100,2])<0.1){ # The last 20 % of the simulation shoul
d reach equilibrium point. I used sd < 0.1 as a threshold to make sure they converge
 to a specific value.
      max_h <- h
      best_period <- p
    }
  }
}
print(paste0("The best period for harvesting the yeast is once every ", p ," day(s)."
))
```

```
## [1] "The best period for harvesting the yeast is once every 1 day(s)."
```

```
print(paste0("The maximal periodic harvesting amount is ", max_h, "."))
```

```
## [1] "The maximal periodic harvesting amount is 1.3."
```

```r
# re-run the optimal simulation
K <- 13
r <- 0.4
h <- max_h
p <- best_period
current_n <- c()
current_n[1] <- K
last_n <- current_n[1]
for (i in 2:100){
  if (i%%p==0){
    current_n[i] = last_n + r*last_n*(1-last_n/K) - h
    last_n <- current_n[i]
  }
  else{
    current_n[i] = last_n + r*last_n*(1-last_n/K)
    last_n <- current_n[i]
  }
}
current_n_df <- cbind(1:100, current_n)
colnames(current_n_df) <- c("Time","Population")
current_n_df <- data.frame(current_n_df)
# plot
q8 <- ggplot(data=as.data.frame(current_n_df),aes(Time,Population)) +
  geom_line() +
  geom_point() +
  ggtitle(paste0("max_h = ", h, "  /  best_period = ", p)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Time (days)") +
  ylim(0, 15)
print(q8)
```


max_h = 1.3  /  best_period = 1
Population
15
10
5
0
0
25
50
75
100
Time (days)