Lab 1: R discrete logistic exercise

Team USA: Yuting Deng, Kendra Gilbertson, Lily Durkee, Bennett Hardy

Hermione.Deng@colostate.edu, kendra01@colostate.edu, L.Durkee@colostate.edu,Bennett.Hardy@colosta

2022-09-06

Exploring chaos with the discrete logistic growth

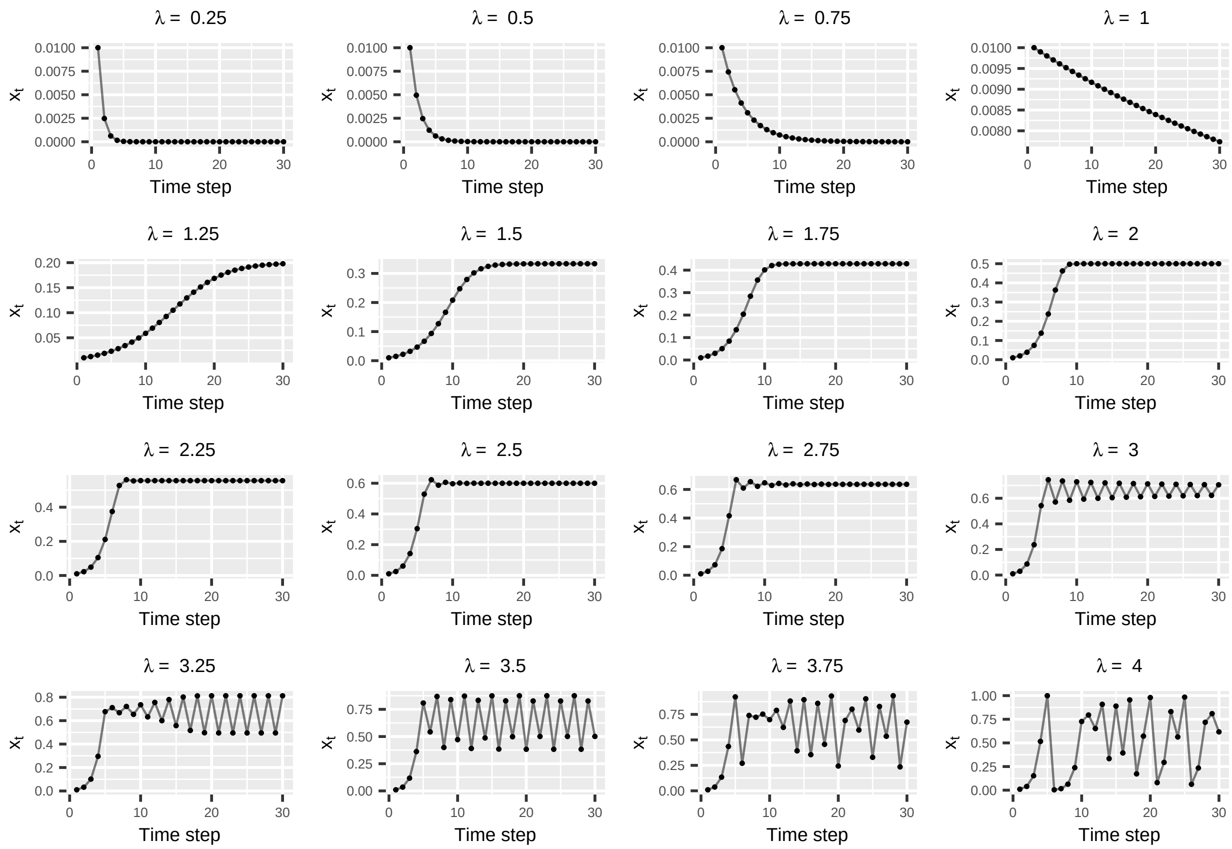

Figure 1: Simulating population size over time in response to λ.

From **Figure 1** we see that the first change in dynamics occurs when $\lambda = 1$, when the population growth rate switches from decreasing ($\lambda > 1$) to increasing ($\lambda < 1$). Then, as λ continues to increase, population size x_t becomes less stable. At $\lambda = 2.75$ we see x_t alternating between two values from approximately $t \geq 7$. The number of values that x_t oscillates between increases to four when $\lambda = 3.5$ (*i.e.,* demonstrating a limit cycle with a period of 4). Then, above 3.5, the oscillations no longer cycle periodically and we see chaos, which is particularly evident when $\lambda = 4$.

In the bifurcation plot in **Figure 2**, we see one value of population size x_t per value of λ, until just before $\lambda = 3$. At this point, we begin to see multiple possible values for x_t, which corresponds to the beginning

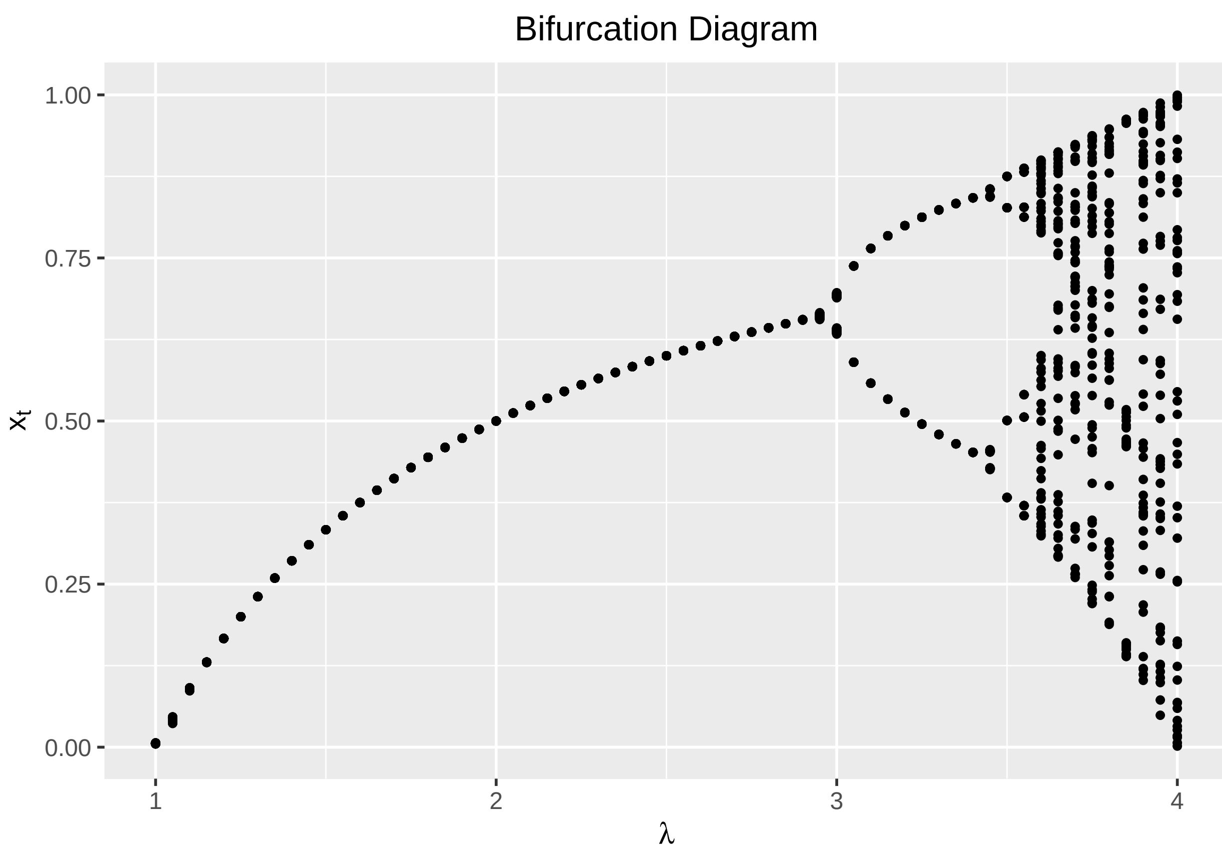

Figure 2: Bifurcation diagram demonstrating the values of x_t as λ increases.

of periodic oscillations that we see in **Figure 1** when $\lambda > 2.5$. As λ continues to increase, the number of possible values for x_t also increases. We see x_t begin oscillating between four values starting around $\lambda = 3.4$, which is reflected in **Figure 1** when $\lambda = 3.5$, and then more than four values starting around $\lambda = 3.6$. When $\lambda > 3.6$, we can see that there are many possible values for x_t, which corresponds to the chaos that we see in the last two plots of **Figure 1**.

Optional, advanced problem: Fitting a logistic model to data by minimizing the sums of squared error

```
## [1] 67164
```

```
##          r   K  N1     SSE
## 67164 0.15 995 395 1169743
```

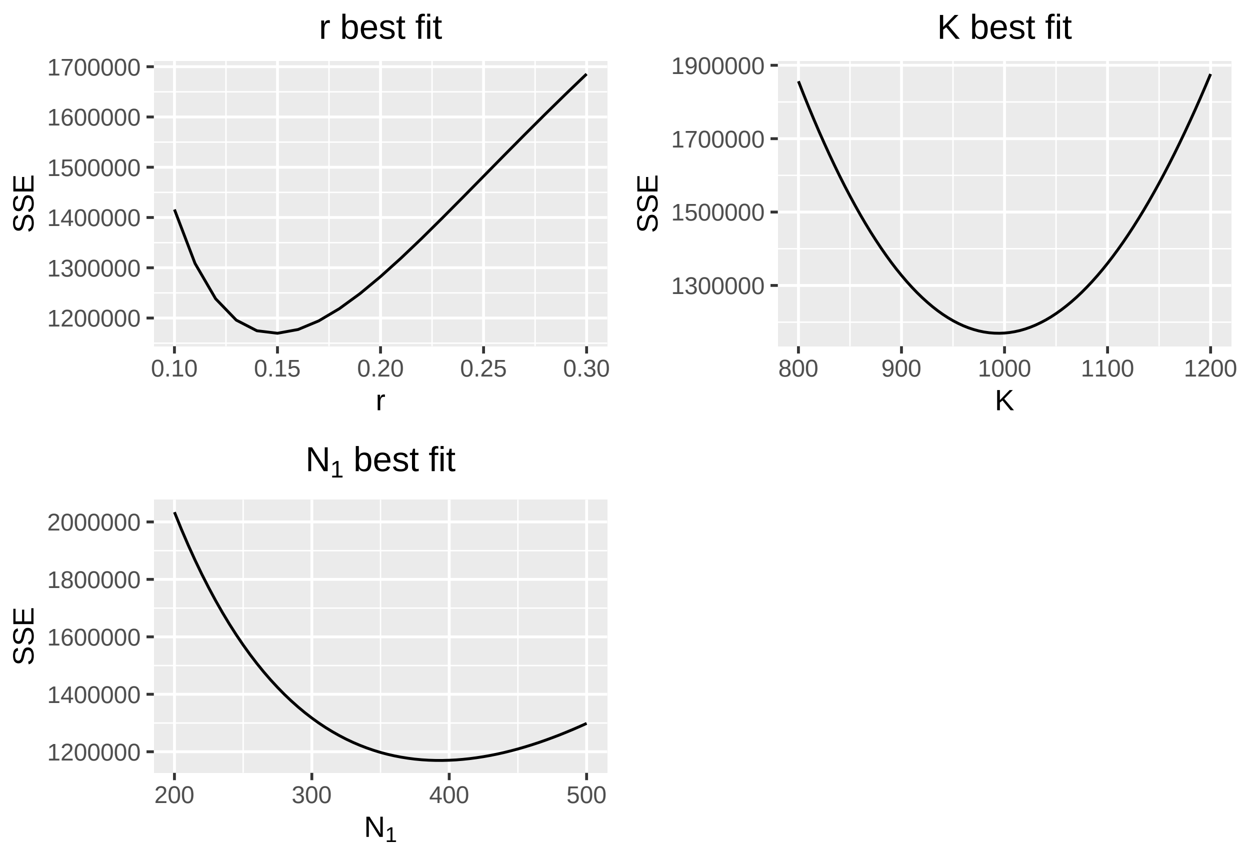

Figure 3: Parameter estimate fit using r, K, and N_1 best fit.

One of the issues with the brute force method is that it's computationally expensive. The simulations will take even longer with more parameter values, and using this method would give you an exponential number of combinations to try. Another issue is that you're testing all possible combinations of parameter values indiscriminately. The result of one combination (whether good or bad) does not affect the values you test next. It would be more efficient to evaluate the fit after each iteration and use that information to inform you next guess. For example, if a combination has a poor fit, we should explore a different area of the sample space, not continue to exhaust resources looking in an area we know is not a good fit. On the other hand, if a combination has good fit, the next iterations should explore that area of sample space to converge around our best fit parameter values.

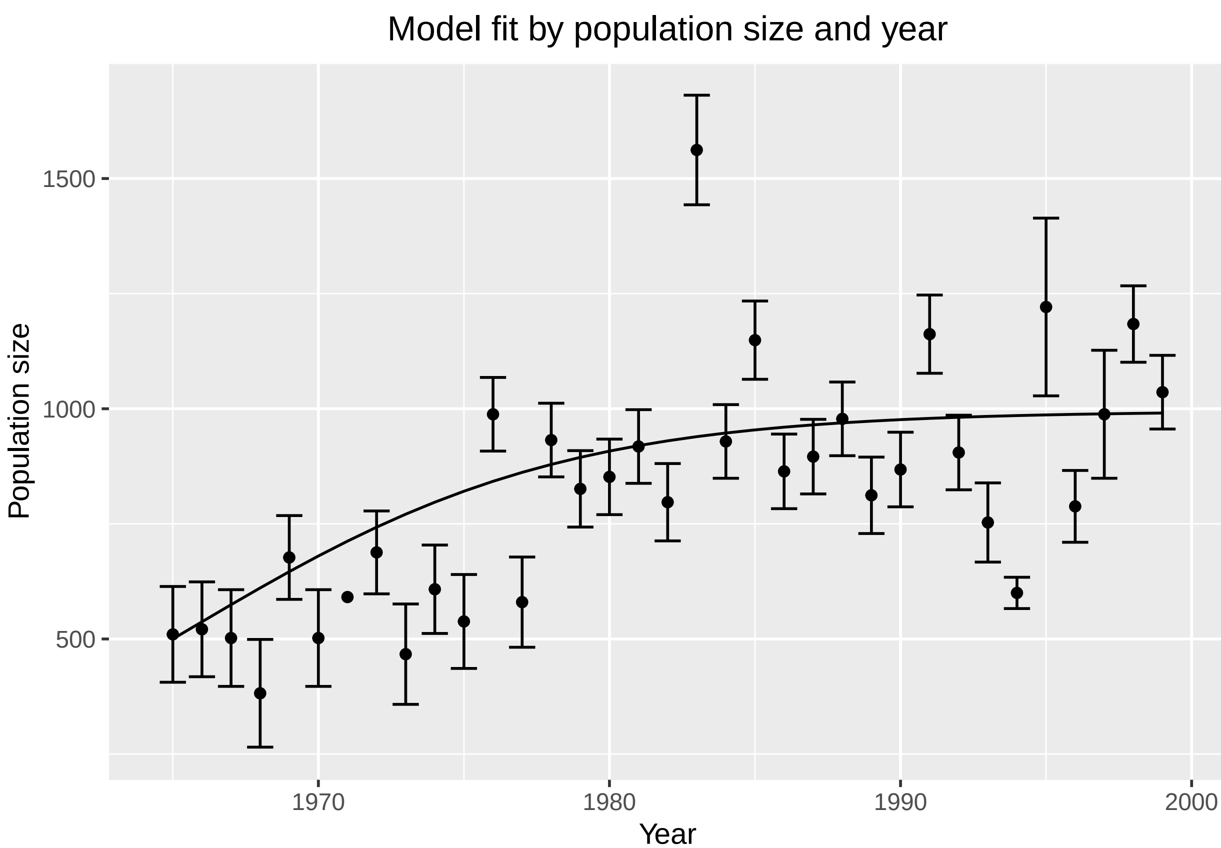

Figure 4: Model fit to the data by population size and year, including one standard deviation.

```r
knitr::opts_chunk$set(echo = FALSE,
  attr.source = '.numberLines',
  warning = FALSE,               # don't show warnings
  message = FALSE                # don't show messages (less serious warnings)
)
library(ggplot2)
library(gridExtra)
library(dplyr)
library(ggpubr)
library(mathjaxr)
#value of growth rate
lambda = seq(0.25, 4, 0.25)
plot_list = list()
i=1
df.2 = data.frame()
for (lam in lambda){
   #Vector for holding state variable
     N = numeric(30)
     #Initial value of N
     N[1] = 0.01
     #Loop to calculate N over time. Use t to index vector.
     for (t in 2:30){
         # store the values of N in a vector
         N[t] = lam * N[t-1]*(1-N[t-1])
     }
     df = data.frame()
     df = data.frame(lambda = rep(lam,30), t=1:30, N = N)
     df.2 = rbind(df.2, df)

     plot_list[[i]] = ggplot()+
       geom_line(df, mapping=aes(x=t, y=N),
                 size=.4, alpha=0.5)+
       geom_point(df, mapping=aes(x=t, y=N),
                 size=.3, alpha=1)+
       labs(title = bquote(lambda ~ "= " ~ .(lam)),x="Time step",y=expression("x"[t]))+
       theme(plot.title = element_text(size=7),axis.title.x = element_text(size=7),
             axis.title.y=element_text(size=7),axis.text=element_text(size=5))+
       theme(plot.title = element_text(hjust = 0.5))
     #  theme_classic()

     i=i+1
}

do.call("grid.arrange", c(plot_list, ncol = 4))

lambda = seq(1, 4, 0.05)
plot_list = list()
i=1
df_2 = data.frame()

for (lam in lambda){
   #Vector for holding state variable
     N = numeric(30)
```

```r
    #Initial value of N
    N[1] = 0.01
    #Loop to calculate N over time. Use t to index vector.
    for (t in 2:100){
        # store the values of N in a vector
        N[t] = lam * N[t-1]*(1-N[t-1])
    }
    df = data.frame(lambda = rep(lam,50), t=51:100, N = N[51:100])
    df_2 = rbind(df_2, df)
}

ggplot()+
       geom_point(df_2, mapping=aes(x=lambda, y=N),
                  size=1, alpha=1)+
       labs(title="Bifurcation Diagram",x=expression(lambda),y=expression("x"[t]))+
       theme(plot.title = element_text(hjust = 0.5))
setwd("/Users/kendragilbertson/Documents/CSU/Classes/ESS 575/Lab 2/Data_for_R_primer")
elk = read.csv("RMNP elk time series.csv")

elk_df = expand.grid(r = seq(0.1, 0.3, 0.01), K = seq(800, 1200, 5), N1 = seq(200,500,5))
elk_df[ , 'SSE'] <- NA

for (i in 1:nrow(elk_df)){
  N = numeric(35)
  r = elk_df[i,1]
  K = elk_df[i,2]
  N[1] = elk_df[i,3]

  for (t in 2:35){
  N[t] = N[t-1] + r * N[t-1]*(1-N[t-1]/K)
}
  elk_df[i,4] = sum((N-elk$Population_size)^2)
}

which (elk_df$SSE == min(elk_df$SSE))
elk_df[which (elk_df$SSE == min(elk_df$SSE)), ]
g1 = ggplot(elk_df %>% filter(K==995 & N1==395))+
  geom_line(aes(x=r, y=SSE))+
  labs(title="r best fit")+
  theme(plot.title = element_text(hjust = 0.5))
   theme_classic()

g2 = ggplot(elk_df %>% filter(N1==395) %>% filter(r == "0.15"))+
  geom_line(aes(x=K, y=SSE))+
    labs(title="K best fit")+
       theme(plot.title = element_text(hjust = 0.5))
  theme_classic()

g3 = ggplot(elk_df %>% filter(r == "0.15" & K==995))+
  geom_line(aes(x=N1, y=SSE))+
  labs(title=expression("N"[1]*" best fit"),x=expression("N"[1]))+
       theme(plot.title = element_text(hjust = 0.5))
  theme_classic()
```

```
ggarrange (g1,g2,g3)
r = 0.15
K = 995
N1 = 395

for (t in 2:35){
  N[t] = N[t-1] + r * N[t-1]*(1-N[t-1]/K)
}
  se = (N-elk$Population_size)^2


df.4 = data.frame(year=1965:1999, N = N, se = se)

ggplot()+
  geom_point(elk, mapping=aes(x=Year, y=Population_size))+
  # geom_pointrange(elk, mapping=aes(x=Year,ymin=Population_size-SE, ymax = Population_size+SE))+
  geom_errorbar(elk, mapping=aes(x=Year,ymin=Population_size-SE, ymax = Population_size+SE))+
  geom_line(df.4, mapping=aes(x=year, y=N))+
  labs(title="Model fit by population size and year",y="Population size")+
   theme(plot.title = element_text(hjust = 0.5))
```