EEB 319 Lab 2- Grizzly Bear Population Dynamics

There are two goals of this lab:

1) To introduce you to R and RStudio, which you will be using throughout this course to analyze data and apply the theoretical ideas learned in class.

2) Analyze the population dynamics of grizzly bears, in particular their risk of extinction.

A quick introduction to R:

To make things clear: R is the name of the programming language while RStudio is a convenient interface. There are many R editors that can be used with R, but RStudio is one of the best ones since it is compatible with both PCs and Macs.

1) Go to www.r-project.org and download the latest version of R that is compatible with your software.

2) Go to rstudio.org and download the latest version that is compatible with your computer

Once RStudio is downloaded and installed, open the application. The panel on the left is known as the *console*. This is where you will be typing commands and interpreting their output. The panel in the upper righthand corner contains your *workspace* and the history of the commands that you have previously entered. In the lower right corner, you will see any plots that you have generated.

Objects

All type of data inputted into R is recognized and saved as "objects." There are 4 types of objects that you should know about:

- Variable: a single value or character
- Vector: a list of values of characters
- Dataframe: a matrix of values
- Function: used to perform an operation in R when provided with an argument

Variables and assignments

To assign a name to a value, use <-. You can assign any name to any variable. For example:

- a=8
- a

[1] 8

- b=4
- b

[1] 4

When doing any type of operation, you can call on the variables that you have assigned using their names. Operations can also be performed using vectors and sequences. For example:

- ab=b/a
- ab

[1] 2

R functions

There are many functions that come with the RStudio function. You will be using the "plot" function frequently throughout these labs to generate plots. In order to use a function, call the function name and use round brackets. For example, to plot your data you would type into the console:

- *plot ( )*

Functions also take arguments or information on which the function must act. For example, if you want to take the natural log of "a", use the function log ( ) where a is the argument of interest:

- *log (a)*

Loops

Loops are useful for running the same operation multiple times such as calculating how a value changes over time. This can be done by writing something called a 'for' loop.

Before writing a for loop, it is important to create an empty vector to store the resulting values in. To then write the for loop, make use of the for() function. Since the loop will be used to run an operation multiple times, it is also necessary to have an index for the current location in the loop. For example, you can use i or t. R also needs to be told where the index is starting and ending. The loop should look something like this:

>result.vector #this is the empty vector to store values in

#This is the loop where the operation will be performed:

>for (t in starting:ending) {

result.vector[t]=operation

}

For example, if we want to take 4 to increasingly higher powers, we would write:

>times=5 #this is the number of times we want the loop to do the operation
>p=seq(3,7,by=1) # this is a vector of powers that we want the loop to take 4 too
>result=rep(NA), length=times #empty vector to store the results in

>for(t in 1:4){

result[t]=4^p[t]

}

>result #prints the results

Let's get started on the lab:

Download the excel file "Bears.xls" from Quercus and save it on your computer as a csv file. It contains a time series of the abundance of adult female grizzly bears in Yellowstone National Park, USA, between 1959 and 1978.

1. (10 points) Plot the abundance of female grizzly bears through time. Use black circles with white fill as your symbol on the plot and connect the symbols with a black line. Remember to label your axes, make your axis labels easily readable, and include a figure caption. Describe in words the trend in abundance.

- To read a file into RStudio, use the function "read.csv"
- To plot the data in the file, use the function "plot."

To read in the grizzly bear data, type into the console the following command:

- data<- read.csv("Bears.csv"), header=TRUE) #read in file and rename it 'data'

2. (10 points) To the datasheet called that you loaded in Question 1, create a new column beside the bear abundances and label it 'lambda.' Under this column, calculate the time series of annual geometric population growth rates, $\lambda_t = N_t+1/N_t$ for each year (note there will be one less λ_t observation than the abundances). Create another new column beside the lambda column and in it calculate the annual exponential population growth rate $r_t = \ln(\lambda_t)$. What is the mean of the exponential population growth rates and what is its standard deviation (hints: use the R functions 'mean' and 'sd').

- Create a for loop to calculate lambda for each year

- To create a new column in the pre-existing spread sheet, use the function 'data.frame' and add the calculated lambda values
- Use the R functions 'mean' and 'sd' to find the mean and standard deviation of lamda

Part 1:

```
Nt <- data[,2] # vector of population abundances
T <- length(Nt) # number of years of data to loop over
lambda <- c() # empty vector to store lamda values in

for (t in 1:T){        #for loop
    lambda[t] <- ((Nt[t+1])/Nt[t])
}

lambda[is.na(lambda)]<-0
lambda #print lambda values

# Add the column to the existing data:

data <- data.frame(data)
data <- data.frame(data$year,data$N,lambda)
```

Part 2:

```
rt<-c()

for (t in 1:T){
    rt[t]<-log(lambda[t])
}

#Add the rt column to the existing data

data <- data.frame(data)
data <- data.frame(data$year,data$N,lambda,rt)
```

3. (15 points) A simple exponential growth model (without stochasticity nor density dependence) for grizzly bear population dynamics is

$$N_{t+1} = N_t \exp(r)$$

where r is the average population growth rate. Using the mean population growth rate r, simulate the model for 50 years using an initial population size of 44 at time t=0. What is the final population size at time t=50? Do it again and determine how long it will take the population to decline to less than 20 bears. Include a plot of the simulation.

A simple stochastic density-independent population model for grizzly bear population dynamics is

$$N_{t+1} = N_t \exp(r + \varepsilon_t),$$

where r is the mean population growth rate and εt random variable from a normal distribution with a mean of zero and standard deviation of the growth rates you calculated in section (1). Recall from lecture that this model has a solution

$$N_t = N_0 \exp(rt + \varphi t),$$

where N_0 is the initial population size, r is the mean exponential population growth rate, and φ_t is a random normal variable with mean=0 and variance = $t\sigma^2$ where the σ^2 is the variance of the r_t values.

Use the exponential growth formula above to relate population size at the current time step relative to the abundance in the previous time step (one cell above it) for 50 time steps using a for loop. Create a new dataframe with two columns. Label the first column 'Time' and the second column 'Abundance' and fill it with your newly calculated values.

```
N0<-44 #initial population size
times=60 #number of years into the future
N<-vector(length=times) #empty vector to store population size in
N[1]=N0 #initial population size should be the first N
r<-(-0.01357) #mean growth rate calculated from question 2

for (t in 2:times){

    N[t]=N[t-1]*exp(r)
 }

df<-data.frame(times,N)
df
plot(1:times,N,type="o",las=1)
```

4. (15 points) In this question we will analyze how the mean and variance of the predicted population size change through time. Plot the predicted mean population size +/- 1 standard deviation from time t=1 to t=50 with a starting population size of 44 at time t=0. Calculate the predicted population size at each time step, one for the mean plus 1 standard deviation, and one for the mean minus one standard deviation using a for loop. Add these values into two new columns in the dataframe that you created in Question 3.

What is happening to the range of uncertainty around the predicted population size as time increases?

#r+sd

```
set.seed(2)
N0<-44
times<-50
N<-vector(length=times)
N[1]=N0
r<-(-0.01357)
sd<-0.0815547

for (t in 2:times){

    N[t]=N[t-1]*exp(r+sd)
}

N
plot(1:times,N,type="o",las=1)
```

#r-sd

```
set.seed(2)
N0<-44
times<-50
N2<-vector(length=times)
N2[1]=N0
r<-(-0.01357)
sd<-0.0815547

for (t in 2:times){

    N2[t]=N2[t-1]*exp(r-sd)
 }

N2
```

```
lines(1:times,N2,type="o",las=1)
lines(1:times,N,type="o",las=1)
```

5. (15 points) For the stochastic model of grizzly bear population dynamics

$N_{t+1} = N_t \exp(r + \varepsilon t)$,

conduct a stochastic simulation of the model from time t=1 to t=50 years with initial population size of 44 at time t=0. (Hint: you can use a similar method to

question (3) but you need to add the random variable at each time step. Include a plot of the simulation and describe what you see in words.

- The function 'rlnorm' will generate a random variable from a distribution with a mean and standard deviation that you specify in R

```
set.seed(2)

N0<-44
times<-50
N<-vector(length=times)
N[1]<-N0

epsilon<-rlnorm(times-1,meanlog=0,sdlog=sd)

for (t in 2:times){

    N[t]=N[t-1]*exp(r+epsilon[t-1])
}

N
plot(1:times,N,type="o",las=1)
```

6. (20 points) Suppose we set 20 bears as a critical minimum population size above which we want to maintain the population. We are interested in estimating the probability that the population will drop below this threshold over a time period of 50 years. Using the model in question (5), estimate this probability by conducting 100 simulations and counting the number of simulations where the population size dips below 20.

    - Create an empty matrix with 100 columns and 50 rows using the 'matrix' function
    - Use the model from Question 5 to simulate the data and save the values into the matrix
    - Use the 'if' function in R to help you identify and count up the simulations that dropped below 20

```
# Loop for get many more randomized epsilons

epsilon_list <- list()
for(i in 1:100){
epsilon_list[[i]] <- rnorm(50,0,1)

}

# run the simulation 100 times

N_list <- list()
```

- for (i in 1:100){
  epsilon <- epsilon_list[[i]]
  N <- c(44,rep(NA,49))
  for (t in 2:time)
  N[t] <- N[t-1]*exp(r + epsilon[t-1])
  N_list[[i]] <- N

}

7. (15 points) For the set of 100 simulations above, calculate the average final population size and the 95% confidence intervals on this estimate.

- Use the 'qnorm' function in R to calculate the 2.5^{th} and 97.5^{th} percentiles of final population sizes