Demographic Modeling Introduction

Matthew E. Aiello-Lammens

Contents

Demographic models and population viability analysis (PVA)

Calculating a population growth rate

```
R <- rnorm(n = 25, mean = 1.05, sd = .15)

n_init <- 1000
n <- vector(length = length(R))

n_old <- n_init
for (t in 1:length(R)){
  n_new <- n_old * R[t]
  n[t] <- n_new
  n_old <- n_new
}

n <- floor(n)
n <- c(n_init, n)

bottlenose_census <- data.frame( year = 1990:2015,
                                 pop_size = n )

write.csv(file = "bottlenose_census.csv", x = bottlenose_census, row.names = FALSE)
```

We will use census data from a multi-year bottlenose dolphin census to calculate annual growth rates. **NOTE: These data are made up for the purposes of this demonstration, and do not represent real population sizes for this species.**

```
bottlenose_census <- read.csv(file = "https://raw.githubusercontent.com/mlammens/DemModel-ShortCourse/ma
```

Let's have a quick look at these data

```
head(bottlenose_census)
```

```
##   year pop_size
## 1 1990     1000
## 2 1991      996
## 3 1992     1185
## 4 1993     1015
## 5 1994      923
## 6 1995     1120
```

```
tail(bottlenose_census)
```

```
##    year pop_size
## 21 2010     2336
## 22 2011     2736
## 23 2012     2873
## 24 2013     3581
## 25 2014     3615
## 26 2015     3292
```

```
summary(bottlenose_census)
```

```
##       year         pop_size
##  Min.   :1990   Min.   : 923
##  1st Qu.:1996   1st Qu.:1130
##  Median :2002   Median :1468
##  Mean   :2002   Mean   :1740
##  3rd Qu.:2009   3rd Qu.:1950
##  Max.   :2015   Max.   :3615
```

Since we have 26 years worth of data, we should be able to calculate 25 R values. Recall the formula for R is:

$$R = \frac{N(t+1)}{N(t)}$$

```
bottlenose_R <- bottlenose_census$pop_size[2:length(bottlenose_census$pop_size)] /
  bottlenose_census$pop_size[1:(length(bottlenose_census$pop_size) - 1)]
```

Let's look at those values by printing them out to the screen and looking at a few summaries.

```
bottlenose_R
```

```
##  [1] 0.9960000 1.1897590 0.8565401 0.9093596 1.2134345 1.1017857 0.8176661
##  [8] 1.0931615 1.0525839 1.0680448 1.2862903 0.9028213 1.0993056 0.9456728
## [15] 1.1349365 1.1571513 0.6846389 1.3268945 1.0643897 1.2288269 1.1712329
## [22] 1.0500731 1.2464323 1.0094946 0.9106501
```

```
min(bottlenose_R)
```

```
## [1] 0.6846389
```

```
max(bottlenose_R)
```

```
## [1] 1.326895
```

Geometric mean of R

```
prod(bottlenose_R)^(1/length(bottlenose_R))
```

```
## [1] 1.048814
```

Standard deviation of R

```
sd(bottlenose_R)
```

```
## [1] 0.1565993
```

Standard error of the mean

```
sd(bottlenose_R)/sqrt(length(bottlenose_R))
```

```
## [1] 0.03131986
```

Simulating population growth

OK. Let's now use this information to simulate the population change for bottlenose dolphins, assuming the past population growth rates are a reasonable representation of future population conditions.

We will use two different methods to do this.

Bootstrap method

First, let's use a bootstrapping method, where we restrict our selves to using only observed R values.

Assume that we want to forecast our population for 25 years. Then we need to draw 25 R values from our current set. As this is bootstrapping for forecasting, we want to sample **with** replacement.

```
bottlenose_R_proj <- sample(bottlenose_R, size = 25, replace = TRUE)
```

Let's look at our new values.

```
bottlenose_R_proj
```

```
##  [1] 0.6846389 1.0993056 0.8565401 0.8176661 1.2288269 1.1349365 1.1017857
##  [8] 0.8565401 0.9456728 0.9028213 1.1017857 1.3268945 1.0500731 1.0643897
## [15] 1.2464323 1.0525839 1.2862903 0.9093596 0.9028213 0.9093596 0.6846389
## [22] 1.2134345 1.2464323 1.0500731 1.0525839
```

```
min(bottlenose_R_proj)
```

```
## [1] 0.6846389
```

```
max(bottlenose_R_proj)
```

```
## [1] 1.326895
```

Geometric mean

```
prod(bottlenose_R_proj)^(1/length(bottlenose_R_proj))
```

```
## [1] 1.013423
```

In order to use these values to forecast the population, we're going to need to use a `for` loop.

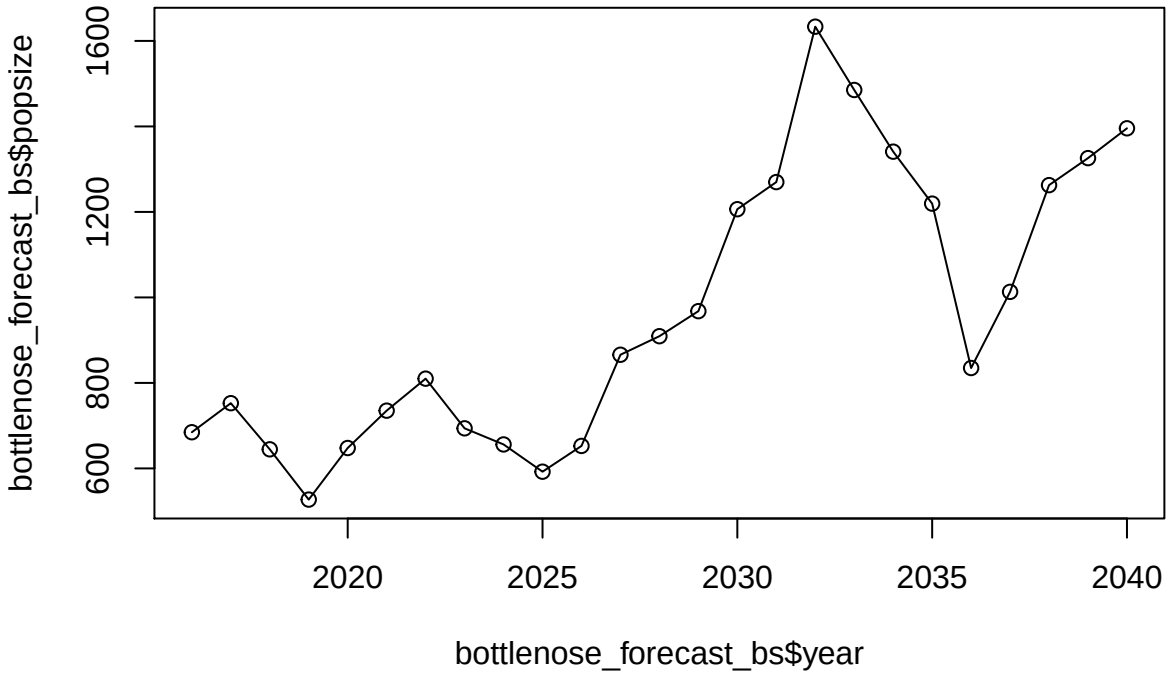
```
# Set our initial value
n_init <- 1000

# Create a vector to store new values in
n <- vector(length = length(bottlenose_R_proj))

n_old <- n_init
for (t in 1:length(bottlenose_R_proj)){
  n_new <- n_old * bottlenose_R_proj[t]
  n[t] <- n_new
  n_old <- n_new
}
```

Make a `data.frame` with these new values

```
bottlenose_forecast_bs <-
  data.frame( year = seq(from = 2016, to = (2016 + length(bottlenose_R_proj) - 1)),
              popsize = n )
```

Plot our values through time

```
plot(x = bottlenose_forecast_bs$year, y = bottlenose_forecast_bs$popsize, type = "o")
```



Calculating extinction risk and other measures

The work above gave us one possible trajectory for this population, but in order to really say something about what we might expect in the future, we need to run this simulation 100s or 1000s of times, and use those values to make probablistic statements.

Let's run 1000 simulations.

First generate 1000 * 25 R values.

```
bottlenose_R_proj_multi <- sample(bottlenose_R, size = (1000*25), replace = TRUE)
```

Make these into a 2D matrix

```
bottlenose_R_proj_multi <- matrix(bottlenose_R_proj_multi, nrow = 25, byrow = FALSE)
```

In order to run these simulations 1000 times, we're going to use a `for` loop inside another `for` loop.

Let's start by making an empty matrix to store all of our simulation results. This matrix needs to have 25 rows and 1000 columns.

```
bottlenose_forecast_bs_multi <- matrix(rep(NA, 1000*25), nrow = 25 )
```

Why did I use NAs here?

Let's make that double `for` loop.

```
# Set our initial value
n_init <- 1000
n_sims <- ncol(bottlenose_forecast_bs_multi)
```

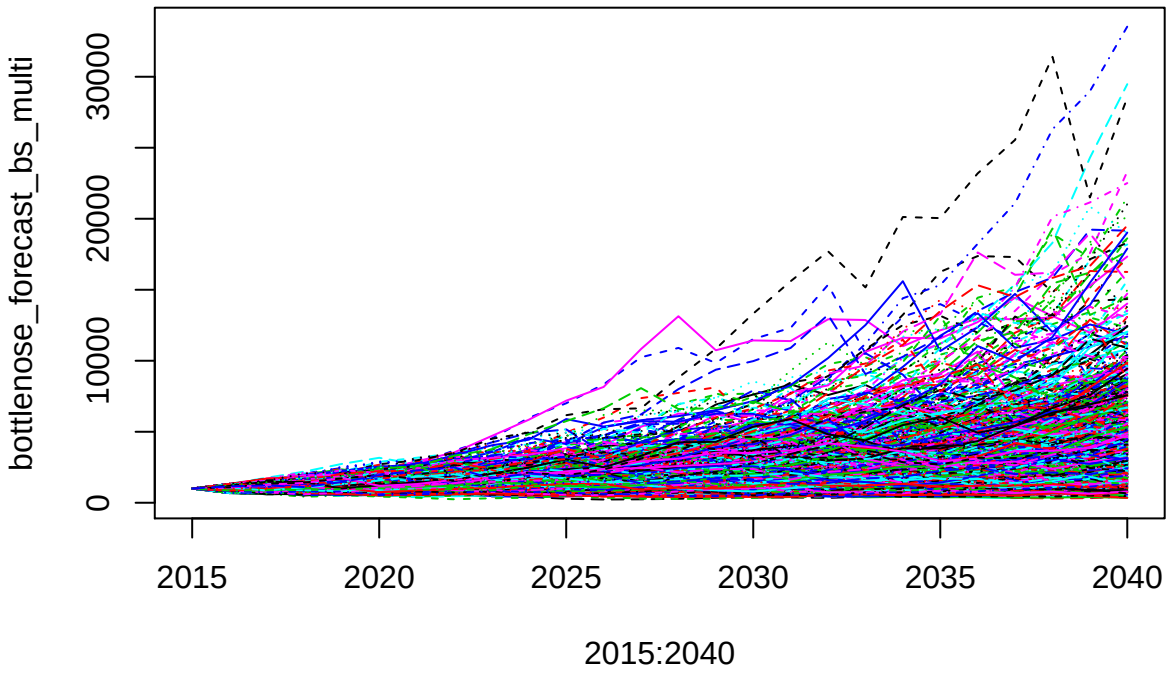
```
for( sim in 1:n_sims){

  # Create a vector to store new values in
  n <- vector(length = nrow(bottlenose_R_proj_multi))

  n_old <- n_init
  for (t in 1:nrow(bottlenose_R_proj_multi)){
    n_new <- n_old * bottlenose_R_proj_multi[t, sim]
    n[t] <- n_new
    n_old <- n_new
  }

  bottlenose_forecast_bs_multi[ , sim] <- n
}
```

Add the initial value as a new **first** row.

```
bottlenose_forecast_bs_multi <- rbind(rep(n_init, n_sims), bottlenose_forecast_bs_multi)
```

Let's plot all of the simulations.

```
matplot(x = 2015:2040, y = bottlenose_forecast_bs_multi, type = "l")
```



We can determine the number of simulations where the population goes extinct by looking for how many populations are 0 individuals after 25 years.

```
sum(bottlenose_forecast_bs_multi[26, ] == 0)
```

```
## [1] 0
```

How about less than some threshold value? Let's say 500.

```r
sum(bottlenose_forecast_bs_multi[26, ] < 700)
```

```
## [1] 27
```

We can use these values to calculate the probability of this event happening. In this case, the probability of decline to below 700 individuals is:

```r
sum(bottlenose_forecast_bs_multi[26, ] < 700) / n_sims
```

```
## [1] 0.027
```

Another useful statistic is the **Expected Minimum Abundance**. Calculating this value will give us a chance to become familiar with a few more R functions.

In order to calculate the EMA, we first need to calculate the minimum value for each simulation.

```r
bottlenose_forecast_min <- apply(X = bottlenose_forecast_bs_multi, MARGIN = 2, FUN = min)
```

Next we can calculate the average minimum value.

```r
mean(bottlenose_forecast_min)
```

```
## [1] 879.838
```

Challange

1. What happens if the population size is much smaller? Say 100 individuals?
2. What happens if boat tours lead to a 0.025 reduction in population growth rate?

Using statistical distribution methods

Instead of bootstrapping, we could parameterize a statistical distribution with the data we've collected.

```r
bottlenose_R_proj <- rlnorm(25, meanlog = log(prod(bottlenose_R)^(1/length(bottlenose_R))), sdlog = sd(l
```

Repeat steps as above

In order to use these values to forecast the population, we're going to need to use a `for` loop.

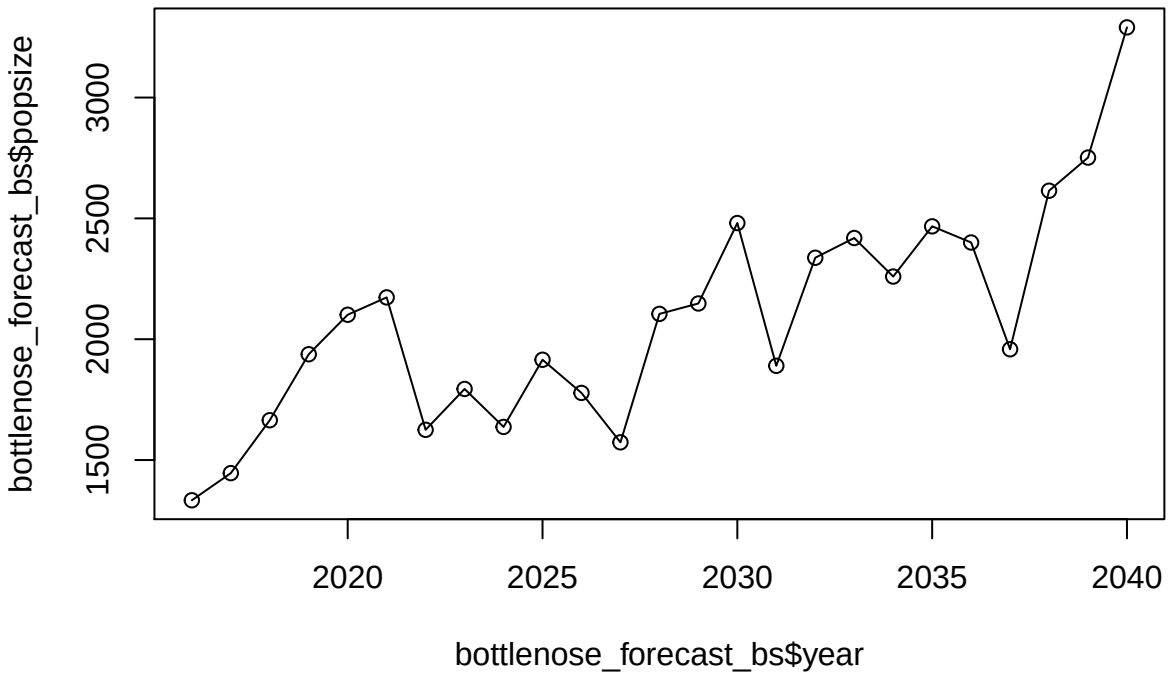
```r
# Set our initial value
n_init <- 1000

# Create a vector to store new values in
n <- vector(length = length(bottlenose_R_proj))

n_old <- n_init
for (t in 1:length(bottlenose_R_proj)){
  n_new <- n_old * bottlenose_R_proj[t]
  n[t] <- n_new
  n_old <- n_new
}
```

Make a `data.frame` with these new values

```
bottlenose_forecast_bs <-
  data.frame( year = seq(from = 2016, to = (2016 + length(bottlenose_R_proj) - 1)),
              popsize = n )
```

Plot our values through time

```
plot(x = bottlenose_forecast_bs$year, y = bottlenose_forecast_bs$popsize, type = "o")
```



Caveats regarding PVA

- Be careful
- Only as good as data
- "All models are wrong, but some are useful" George Box

Matrix projection models

Considering other sources of variability