Tortoise Occupancy

Ellen Bledsoe

2023-04-03

Desert Tortoise Occupancy

Analyze survey data for desert tortoises from Saguaro National Park

Set-Up

I have installed the `unmarked` package for you already in Posit Cloud. This is a new package we will be using to do our occupancy calculations.

The code for using `unmarked` is not intuitive AT ALL. We will walk through an example together first, and then you can apply the code we write today to your assignment.

We will need to first load the `tidyverse` and `unmarked` packages.

```
library(tidyverse)
library(unmarked)
```

Now we need to read in our data set, called `tortoise_occupancy.csv`

```
tort_occ <- read_csv("../data_raw/tortoise_occupancy.csv")
```

```
## Rows: 20 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (5): j_1, j_2, j_3, j_4, j_5
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Explore the data

First, let's take a look at `tort_occ`, our tortoise occupancy data. What does each column represent? How about each row?

```
head(tort_occ)
```

```
## # A tibble: 6 x 5
##     j_1   j_2   j_3   j_4   j_5
##   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1      1      1      1      1      1
## 2      0      0      1      0      0
## 3      0      0      0      0      0
## 4      1      0      1      0      0
## 5      0      0      0      0      0
## 6      1      1      0      0      0
```

Naive Occupancy

Before we run any occupancy models that take detection probability into account, let's calculate the naive occupancy.

Naive occupancy is calculated by the number of sites where a species was detected (at least once) divided by the total number of sites surveyed.

```
sites_detected <- tort_occ %>%
  # count up the number of times each site was marked as occupied
  mutate(detected = rowSums(.))
sites_detected
```

```
## # A tibble: 20 x 6
##      j_1   j_2   j_3   j_4   j_5 detected
##    <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>
##  1     1     1     1     1     1        5
##  2     0     0     1     0     0        1
##  3     0     0     0     0     0        0
##  4     1     0     1     0     0        2
##  5     0     0     0     0     0        0
##  6     1     1     0     0     0        2
##  7     1     1     0     1     1        4
##  8     0     0     0     0     0        0
##  9     0     0     0     0     0        0
## 10     1     0     1     1     0        3
## 11     0     0     1     0     0        1
## 12     1     1     1     0     1        4
## 13     0     0     0     0     0        0
## 14     0     0     1     0     1        2
## 15     0     0     0     0     0        0
## 16     0     0     0     0     0        0
## 17     0     0     0     0     0        0
## 18     1     0     0     1     1        3
## 19     1     0     0     1     0        2
## 20     1     0     1     0     1        3
```

```
naive_occupancy <- sites_detected %>%
            # total number of sites that were surveyed
  summarise(s = n(),
            # number of sites with detection of greater than zero
            s_detected = sum(detected > 0)) %>%
  # divide the number of occupied sites by the total number of sites
  mutate(naive_occupancy = s_detected / s)
naive_occupancy
```

```
## # A tibble: 1 x 3
##       s s_detected naive_occupancy
##   <int>      <int>           <dbl>
## 1    20         12             0.6
```

Is this estimate of occupancy going to be higher or lower than when we take detection probability into account?

Occupancy Models

Now we are going to run a few models that take detection probability into account in our estimate. To do this, we will use the `unmarked` package, which has functions specifically for running these types of models.

To run correctly, the `unmarked` package requires we put our data in a very specific format.

Prepare Our Data

Create a time data frame First, we need to create a list of data frames (why, I don't know…) that includes our survey occasions.

```
# create the data frame with 20 rows and each column with the survey number
surveys <- tibble(.rows = 20,
                  time_1 = "1",
                  time_2 = "2",
                  time_3 = "3",
                  time_4 = "4",
                  time_5 = "5")

# make this into a list of one dataframe
surveys <- list(time = surveys)
```

Create `unmarked` data frame Now that we have both our occupancy data frame (the encounter histories) and a list that contains our survey data frame, we can create the specific type of data frame that the `unmarked` package requires. To do this, we use the `unmarkedFrameOccu()` function.

Running this line of code will produce a warning, but you can ignore it.

```
tortUMF <- unmarkedFrameOccu(tort_occ, obsCovs = surveys)
```

```
## Warning: obsCovs contains characters. Converting them to factors.
```

Let's take a look at this new data frame that we just created.

```
head(tortUMF)
```

```
## Data frame representation of unmarkedFrame object.
##    y.1 y.2 y.3 y.4 y.5 time.1 time.2 time.3 time.4 time.5
## 1    1   1   1   1   1      1      2      3      4      5
## 2    0   0   1   0   0      1      2      3      4      5
## 3    0   0   0   0   0      1      2      3      4      5
## 4    1   0   1   0   0      1      2      3      4      5
## 5    0   0   0   0   0       1       2       3       4       5
## 6    1   1   0   0   0       1       2       3       4       5
## 7    1   1   0   1   1       1       2       3       4       5
## 8    0   0   0   0   0       1       2       3       4       5
## 9    0   0   0   0   0       1       2       3       4       5
## 10   1   0   1   1   0       1       2       3       4       5
```

Fit Models for Occupancy

Occupancy models take the data from our encounter histories and first estimates detection probability. It then uses that detection probability estimate to produce an estimate of occupancy (our parameter of interest).

The function to run the occupancy model in `unmarked` is `occu()`. The function always take the argument for detection probability first and then the argument for occupancy.

Let's demonstrate. Our first model will have detection probability consistent across time (surveys) and occupancy consistent across sites. Remember, an assumption we have about occupancy is that it stays consistent through our surveys.

```
# Model p(.)psi(.) -- psi constant across sites, det prob constant over time
pdot_psidot  <- occu(~1 ~1, data = tortUMF)
```

Our second model will have occupancy consistent across sites but detection variability can vary with time (survey occasion).

```
# Model p(t)psi(.) -- psi constant across sites, det prob varies with time (survey occasion)
ptime_psidot <- occu(~time-1 ~1, data = tortUMF)
```

Model Selection

We can run both models and then use AIC (an index of model quality) to determine which model fits our data the best. We want to choose the model with the lowest AIC because it represents the most parsimonious model.

We first make an object with a list of our models. Then we have the function `modSel` run the models and present us with the AIC values.

```
# Assemble models into a list to compare their AIC values
models <- fitList(pdot_psidot  = pdot_psidot,
                  ptime_psidot = ptime_psidot)
modSel(models)
```

```
##              nPars    AIC delta AICwt cumltvWt
## pdot_psidot      2 113.25  0.00  0.75     0.75
## ptime_psidot     6 115.49  2.23  0.25     1.00
```

Based on the AIC values, which is our top model?

Estimates & 95% CIs

Now that we have chosen our top model, we want to discover what the actual values detection probability (p) and occupancy (psi) are and our uncertainty around them (95% confidence intervals).

Thankfully, there are functions that produce these values for us.

First, let's look at the parameter estimates (p and psi)

```
# This pulls out the coefficients (estimates) for the model

# For model: p(.)psi(.)
plogis(coef(pdot_psidot))  # estimates of psi and p
```

```
##  psi(Int)    p(Int)
## 0.6157371 0.5197018
```

```
# For model:  p(t)psi(.)
plogis(coef(ptime_psidot))
```

```
##  psi(Int)  p(time1)  p(time2)  p(time3)  p(time4)  p(time5)
## 0.6112910 0.7361347 0.3271639 0.6543438 0.4089494 0.4907800
```

Note that there are 5 estimates for detection probability in the model that we allowed to vary through time. This is because the model produced a detection probability estimate for each survey occasion rather than just one overall detection probability estimate.

We can also calculate the 95% confidence intervals for each estimate. This time we use the `confint` function. We need to indicate the estimate type:

- detection probability: `type = "det"`
- occupancy: `type = "state"` , as is the "true state" of occupancy

```
# model p(.)psi(.)
plogis(confint(pdot_psidot, type = "det",   level = 0.95)) # 95% CI for p
```

```
##            0.025     0.975
## p(Int) 0.3867842 0.6498879
```

```
plogis(confint(pdot_psidot, type = "state", level = 0.95)) # 95% CI for psi
```

```
##              0.025     0.975
## psi(Int) 0.3858832 0.8033929
```

```
# model p(t)psi(.)
plogis(confint(ptime_psidot, type = "det",   level = 0.95))
```

```
##              0.025     0.975
## p(time1) 0.4295640 0.9117811
## p(time2) 0.1276352 0.6177361
## p(time3) 0.3625520 0.8630293
## p(time4) 0.1799118 0.6857501
## p(time5) 0.2368970 0.7495098
```

```
plogis(confint(ptime_psidot, type = "state", level = 0.95))
```

```
##              0.025     0.975
## psi(Int) 0.3843875 0.7984203
```

Similarly to the parameter estimates, we have 5 sets of confidence intervals for the model that allows the estimates of detection probability to vary with time; these are the confidence intervals for each estimate.