Overview of Unmarked:
An R Package for the Analysis of Data from Unmarked Animals

Ian Fiske and Richard Chandler
June 30, 2020

Abstract

Unmarked aims to be a complete environment for the statistical analysis of data from surveys of unmarked animals. Currently, the focus is on hierarchical models that separately model a latent state (or states) and an observation process. This vignette provides a brief overview of the package — for a more thorough treatment see [2]

1 Overview of unmarked

Unmarked provides methods to estimate site occupancy, abundance, and density of animals (or possibly other organisms/objects) that cannot be detected with certainty. Numerous models are available that correspond to specialized survey methods such as temporally replicated surveys, distance sampling, removal sampling, and double observer sampling. These data are often associated with metadata related to the design of the study. For example, in distance sampling, the study design (line- or point-transect), distance class break points, transect lengths, and units of measurement need to be accounted for in the analysis. Unmarked uses S4 classes to store data and metadata in a way that allows for easy data manipulation, summarization, and model specification. Table 1 lists the currently implemented models and their associated fitting functions and data classes.

<table>
<thead>
<tr>
<th>Model</th>
<th>Fitting Function</th>
<th>Data Class</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occupancy</td>
<td>occu</td>
<td>unmarkedFrameOccu</td>
<td>[4]</td>
</tr>
<tr>
<td>Royle-Nichols</td>
<td>occuRN</td>
<td>unmarkedFrameOccu</td>
<td>[8]</td>
</tr>
<tr>
<td>Point Count</td>
<td>pcount</td>
<td>unmarkedFramePCount</td>
<td>[6]</td>
</tr>
<tr>
<td>Distance-sampling</td>
<td>distsamp</td>
<td>unmarkedFrameDS</td>
<td>[7]</td>
</tr>
<tr>
<td>Generalized distance-sampling</td>
<td>gdistsamp</td>
<td>unmarkedFrameGDS</td>
<td>[1]</td>
</tr>
<tr>
<td>Arbitrary multinomial-Poisson</td>
<td>multinomPois</td>
<td>unmarkedFrameMPois</td>
<td>[5]</td>
</tr>
<tr>
<td>Colonization-extinction</td>
<td>colExt</td>
<td>unmarkedMultFrame</td>
<td>[3]</td>
</tr>
<tr>
<td>Generalized multinomial-mixture</td>
<td>gmultmix</td>
<td>unmarkedFrameGMM</td>
<td>[5]</td>
</tr>
</tbody>
</table>

Table 1: Models handled by unmarked.

Each data class can be created with a call to the constructor function of the same name as described in the examples below.

2 Typical unmarked session

The first step is to import the data into R, which we do below using the \texttt{read.csv} function. Next, the data need to be formatted for use with a specific model fitting function. This can be accomplished with a call to the appropriate type of \texttt{unmarkedFrame}. For example, to prepare the data for a single-season site-occupancy analysis, the function \texttt{unmarkedFrameOccu} is used.

2.1 Importing and formatting data

```r
> library(unmarked)
> wt <- read.csv(system.file("csv", "widewt.csv", package="unmarked"))
> y <- wt[,2:4]
> siteCovs <- wt[,c("elev", "forest", "length")]
> obsCovs <- list(date=wt[,c("date.1", "date.2", "date.3")],
```
### 2.2 Fitting models

Occupancy models can then be fit with the `occu()` function:

```r
> fm1 <- occu(~1, pcru)
> fm2 <- occu(~ MinAfterSunset + Temperature ~ 1, pcru)
> fm2
```

**Call:**
```
occu(formula = ~MinAfterSunset + Temperature ~ 1, data = pcru)
```

**Occupancy:**

```
Estimate  SE   z  P(>|z|)
 1.54  0.292  5.26 1.42e-07
```

**Detection:**

```
Estimate  SE   z  P(>|z|)
(Intercept)  0.2098  0.206  1.017 3.09e-01
```
Here, we have specified that the detection process is modeled with the MinAfterSunset and Temperature covariates. No covariates are specified for occupancy here. See ?occu for more details.

2.3 Back-transforming parameter estimates

Unmarked fitting functions return unmarkedFit objects which can be queried to investigate the model fit. Variables can be back-transformed to the unconstrained scale using backTransform. Standard errors are computed using the delta method.

```r
> backTransform(fm2, 'state')
Backtransformed linear combination(s) of Occupancy estimate(s)

  Estimate    SE  LinComb (Intercept) MinAfterSunset Temperature
      0.823 0.042 1.54                1

Transformation: logistic
```

The expected probability that a site was occupied is 0.823. This estimate applies to the hypothetical population of all possible sites, not the sites found in our sample. For a good discussion of population-level vs finite-sample inference, see Royle and Dorazio [9] page 117. Note also that finite-sample quantities can be computed in unmarked using empirical Bayes methods as demonstrated at the end of this document.

Back-transforming the estimate of \( \psi \) was easy because there were no covariates. Because the detection component was modeled with covariates, \( p \) is a function, not just a scalar quantity, and so we need to provide values of our covariates to obtain an estimate of \( p \). Here, we request the probability of detection given a site is occupied and all covariates are set to 0.

```r
> backTransform(linearComb(fm2, coefficients = c(1,0,0), type = 'det'))
Backtransformed linear combination(s) of Detection estimate(s)

  Estimate    SE  LinComb (Intercept) MinAfterSunset Temperature
      0.552 0.051 0.21                1 0.0

Transformation: logistic
```

Thus, we can say that the expected probability of detection was 0.552 when time of day and temperature are fixed at their mean value. A predict method also exists, which can be used to obtain estimates of parameters at specific covariate values.

```r
> newData <- data.frame(MinAfterSunset = 0, Temperature = -2:2)
> round(predict(fm2, type = 'det', newdata = newData, appendData=TRUE), 2)
Predicted    SE  lower upper MinAfterSunset Temperature
 1 0.98 0.01 0.93 1.00 0 0.0 0
 2 0.89 0.04 0.80 0.95 0 0.0 1
 3 0.55 0.05 0.46 0.65 0 0.0 2
 4 0.16 0.03 0.10 0.23 0 0.0 2
 5 0.03 0.01 0.01 0.07 0 0.0 2

Confidence intervals are requested with confint, using either the asymptotic normal approximation or profiling.

```r
> confint(fm2, type='det')
p(Int)         -0.1946871 0.6142292
p(MinAfterSunset) -0.3985642 0.2274722
p(Temperature)   -2.4638797 -1.3233511
> confint(fm2, type='det', method = "profile")
Profiling parameter 1 of 3 ... done.
Profiling parameter 2 of 3 ... done.
Profiling parameter 3 of 3 ... done.

0.025 0.975
```
The output of this code example shows the following:

1. The model selection process using the `fitList` function, which organizes models for further analysis.
2. The multi-model inference using the `modSel` function to select the best model.
3. The prediction of the model using the `predict` function with the `type='det'` option.
4. The parametric bootstrap for model checking using the `parboot` function.

The parametric bootstrap can be used to check the adequacy of model fit. Here we use a χ² statistic appropriate for binary data.

The derived parameters and empirical Bayes methods section explains how to compute confidence intervals for derived parameters using the `parboot` function and the `ranef` function from the `unmarked` package.
> re <- ranef(fm2)
> EBUP <- bup(re, stat="mode")
> CI <- confint(re, level=0.9)
> rbind(PAO = c(Estimate = sum(EBUP), colSums(CI)) / 130)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate</td>
<td>0.8076923</td>
</tr>
<tr>
<td>5%</td>
<td>0.7384615</td>
</tr>
<tr>
<td>95%</td>
<td>0.9929077</td>
</tr>
</tbody>
</table>

Note that this is similar, but slightly lower than the population-level estimate of \( \psi \) obtained above.

A plot method also exists for objects returned by `ranef`, but distributions of binary variables are not so pretty. Try it out on a fitted abundance model instead.

References


