Overview of Unmarked:
An R Package for the Analysis of Data from Unmarked Animals
Ian Fiske and Richard Chandler
March 4, 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</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>gdsamp</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>coext</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 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 unmarkedFrame. For example, to prepare the data for a single-season site-occupancy analysis, the function 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")],
```
ivel <- wt[, c("ivel.1", "ivel.2", "ivel.3")]
wt <- unmarkedFrameOccu(y = y, siteCovs = siteCovs, obsCovs = obsCovs)
summary(wt)

unmarkedFrame Object

237 sites
Maximum number of observations per site: 3
Mean number of observations per site: 2.81
Sites with at least one detection: 79

Tabulation of y observations:
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>&lt;NA&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>483</td>
<td>182</td>
<td>46</td>
</tr>
</tbody>
</table>

Site-level covariates:

<table>
<thead>
<tr>
<th>elev</th>
<th>forest</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : -1.436125</td>
<td>Min. : -1.265352</td>
<td>Min. : 0.1823</td>
</tr>
<tr>
<td>1st Qu. : -0.940726</td>
<td>1st Qu. : -0.974355</td>
<td>1st Qu. : 1.4351</td>
</tr>
<tr>
<td>Median : 0.007612</td>
<td>Median : 0.000888</td>
<td>Median : 1.5924</td>
</tr>
<tr>
<td>Mean : 0.007612</td>
<td>Mean : 0.000888</td>
<td>Mean : 1.5924</td>
</tr>
<tr>
<td>3rd Qu. : 0.994425</td>
<td>3rd Qu. : 0.808005</td>
<td>3rd Qu. : 1.7750</td>
</tr>
<tr>
<td>Max. : 2.434177</td>
<td>Max. : 2.299367</td>
<td>Max. : 2.2407</td>
</tr>
</tbody>
</table>

Observation-level covariates:

<table>
<thead>
<tr>
<th>date</th>
<th>ivel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : -2.90434</td>
<td>Min. : -1.7533</td>
</tr>
<tr>
<td>1st Qu. : -1.11862</td>
<td>1st Qu. : -0.6660</td>
</tr>
<tr>
<td>Median : -0.11862</td>
<td>Median : -0.1395</td>
</tr>
<tr>
<td>Mean : -0.00022</td>
<td>Mean : 0.0000</td>
</tr>
<tr>
<td>3rd Qu. : 1.30995</td>
<td>3rd Qu. : 0.8493</td>
</tr>
<tr>
<td>Max. : 3.80995</td>
<td>Max. : 5.9795</td>
</tr>
<tr>
<td>NA's : 42</td>
<td>NA's : 46</td>
</tr>
</tbody>
</table>

Alternatively, the convenience function csvToUMF can be used

> wt <- csvToUMF(system.file("csv", "widewt.csv", package="unmarked"),
               long = FALSE, type = "unmarkedFrameOccu")

If not all sites have the same numbers of observations, then manual importation of data in long format can be tricky. csvToUMF seamlessly handles this situation.

> pcru <- csvToUMF(system.file("csv", "frog2001pcru.csv", package="unmarked"),
                  long = TRUE, type = "unmarkedFrameOccu")

To help stabilize the numerical optimization algorithm, we recommend standardizing the covariates.

> obsCovs(pcru) <- scale(obsCovs(pcru))

2.2 Fitting models

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

> fm1 <- occu(~1 ~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 |
MinAfterSunset  -0.0855  0.160  -0.536  5.92e-01
Temperature     -1.8936  0.291  -6.508  7.60e-11

AIC: 356.7591

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)
0.823     0.0425    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     0.0510    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 -2
2 0.89 0.04 0.78 0.95 0 -1
3 0.56 0.05 0.46 0.65 0 0
4 0.16 0.03 0.10 0.23 0 1
5 0.03 0.01 0.01 0.07 0 2

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

```r
> confint(fm2, type='det')
       2.5  97.5
p(Int)   0.025 0.975
p(MinAfterSunset)  -0.195 0.614
p(Temperature) -2.464 -1.323

> confint(fm2, type='det', method = "profile")
Profiling parameter 1 of 3 ... done.
Profiling parameter 2 of 3 ... done.
Profiling parameter 3 of 3 ... done.
           2.5     97.5
0.025 0.025 0.975 0.975
p(Int)  -0.1929210  0.6208837
p(MinAfterSunset)  -0.4044794  0.2244221
p(Temperature)  -2.5189984  -1.3789261

2.4 Model selection and model fit

Model selection and multi-model inference can be implemented after organizing models using the fitList function.

```r
> fms <- fitList('psi(.)p(.) = fm1, 'psi(.)p(Time+Temp) = fm2)
> modSel(fms)
```

<table>
<thead>
<tr>
<th>nPars</th>
<th>AIC</th>
<th>delta</th>
<th>AICwt</th>
<th>cumlvWt</th>
</tr>
</thead>
<tbody>
<tr>
<td>psi(.)p(Time+Temp)</td>
<td>4</td>
<td>356.76</td>
<td>0.00</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>psi(.)p(.)</td>
<td>2</td>
<td>461.00</td>
<td>104.25</td>
<td>2.3e-23</td>
</tr>
</tbody>
</table>

```r
> predict(fms, type='det', newdata = newData)
```

<table>
<thead>
<tr>
<th>Predicted</th>
<th>SE</th>
<th>lower</th>
<th>upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.98196076</td>
<td>0.01266193</td>
<td>0.9306044</td>
</tr>
<tr>
<td>2</td>
<td>0.89123189</td>
<td>0.04248804</td>
<td>0.7763166</td>
</tr>
<tr>
<td>3</td>
<td>0.55225129</td>
<td>0.05102660</td>
<td>0.4514814</td>
</tr>
<tr>
<td>4</td>
<td>0.15658708</td>
<td>0.03298276</td>
<td>0.1021713</td>
</tr>
<tr>
<td>5</td>
<td>0.02718682</td>
<td>0.01326263</td>
<td>0.0103505</td>
</tr>
</tbody>
</table>

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

```r
> chisq <- function(fm) {
  umf <- getData(fm)
  y <- getY(umf)
  y[y>1] <- 1
  sr <- fm@sitesRemoved
  if(length(sr)>0)
    y <- y[-sr,,drop=FALSE]
  fv <- fitted(fm, na.rm=TRUE)
  y[is.na(fv)] <- NA
  sum((y-fv)^2/(fv*(1-fv)), na.rm=TRUE)
}
> (pb <- parboot(fm2, statistic=chisq, nsim=100, parallel=FALSE))
```

Call:  
parboot(object = fm2, statistic = chisq, nsim = 100, parallel = FALSE)

Parametric Bootstrap Statistics:

<table>
<thead>
<tr>
<th>t0</th>
<th>mean(t0 - t_B)</th>
<th>StdDev(t0 - t_B)</th>
<th>Pr(t_B &gt; t0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>356</td>
<td>20.2</td>
<td>15.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t_B quantiles:</th>
<th>0%</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>t*B</td>
<td>299</td>
<td>306</td>
<td>326</td>
<td>334</td>
<td>346</td>
<td>371</td>
<td>385</td>
</tr>
</tbody>
</table>

| t0 = Original statistic computed from data |
| t_B = Vector of bootstrap samples |

We fail to reject the null hypothesis, and conclude that the model fit is adequate.

2.5 Derived parameters and empirical Bayes methods

The parboot function can be also be used to compute confidence intervals for estimates of derived parameters, such as the proportion of sites occupied PAO = \sum z_i, where z_i is the true occurrence state at site i, which is unknown at sites where no individuals were detected. The “colext” vignette shows examples of using parboot to obtain confidence intervals for such derived quantities. An alternative way achieving this goal is to use empirical Bayes methods, which were introduced in unmarked version 0.9-5. These methods estimate the posterior distribution of the latent variable given the data and the estimates of the fixed effects (the MLEs). The mean or the mode of the estimated posterior distribution is referred to as the empirical best unbiased predictor (EBUP), which in unmarked can be obtained by applying the bup function to the estimates of the posterior distributions returned by the ranef function. The following code returns the estimate of PAO and a 90% confidence interval.
> 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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate</td>
<td>0.7076923</td>
<td>0.7384615</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


