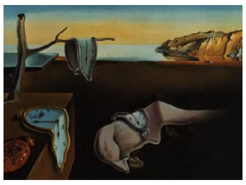

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

unmarked: An R package for the Analysis of Wildlife Occurrence and Abundance Data

Ian J. Fiske
North Carolina State University

Richard Chandler
USGS Patuxent Wildlife Research Center

Abstract

Ecological research uses data collection techniques that are prone to substantial and unique types of measurement error to address scientific questions about species abundance and distribution. These data collection schemes include a number of survey methods in which unmarked individuals are counted, or determined to be present, at spatially-referenced sites. Examples include site occupancy sampling, repeated counts, distance sampling, removal sampling, and double observer sampling. To appropriately analyze these data, hierarchical models have been developed to separately model explanatory variables of both a latent abundance or occurrence process and a conditional detection process. Because these models have a straightforward interpretation paralleling mechanisms under which the data arose, they have recently gained immense popularity. The common hierarchical structure of these models is well-suited for a unified modeling interface. The R package **unmarked** provides such a unified modeling framework, including tools for data exploration, model fitting, model criticism, post-hoc analysis, and model comparison.

Keywords: ecological, wildlife, hierarchical, occupancy, occurrence, distance, point count.

1. Introduction

1.1. Imperfect detection in ecological research

A fundamental goal of ecological research is to understand how environmental variables influence spatial or temporal variation in species abundance or occurrence. Addressing these research questions is complicated by imperfect detection of individuals or species. For example, a species may go undetected when present for a variety of reasons including its proximity to the observer, cryptic behavior, or camouflage. Historically, ecologists ignored the detection process and fit standard generalized linear models to such data; however, failure to account for imperfect detection can yield grossly biased estimates of abundance or occurrence.

To explicitly accommodate imperfect detection of individuals or species, ecologists have developed specialized methods to survey wildlife populations such as site occupancy sampling, repeated counts, distance sampling, removal sampling, and double observer sampling (see Section 2 and (Williams *et al.* 2002) for definitions). This myriad of sampling methods are all unified by a common repeated-measures type of sampling design in which each of a number of spatial sampling units is sampled repeatedly. Similarly, data arising from studies based on such protocols give rise to natural hierarchical models which recognize that observations are generated by a combination of (1) a *state* process determining species abundance at each site and (2) a *detection* process that yields observations conditional on the abundance. Within this repeated measures context, 'within site' variation is informative about factors influencing uncertain observation of the state process, and 'among site' variation is informative about factors that influence the underlying abundance or occupancy state variable.

While many ecological studies produce data consistent with a repeated measures type of sampling design, the development and implementation of hierarchical models for inference under individual sampling protocols has proceeded in a piecemeal fashion, without a unified and coherent analysis platform or using methods that are largely inaccessible to practitioners ... [maybe use that as a segue into this sentence.] This paper introduces **unmarked**, an R package that provides a unified approach for likelihood analysis of many classes of hierarchical models that have been developed for sampling biological populations.

We note that another broad-class of models relevant to ecological studies are so-called 'capture-recapture' models. Such models are widely used in studies of populations in which individuals can be uniquely marked and recaptured (and identified) during subsequent sampling. Further, such models are normally applied to only one or a small number of geographically cohesive populations in which spatial sampling is not usually an explicit element of the model. A huge number of capture-recapture and related methods is available in the software package MARK (Burnham and White XXXX; Cooch book). The origins of **unmarked** (the package and its name) arose to fill the void of models for data from studies of unmarked individuals involving explicit spatial sampling of units.

1.2. Hierarchical Models for Metapopulation Designs

I suggest a short section 1 or 2 pars where you say: (1) the data structure common to basically all functional elements of unmarked is the repeated measures design. Sites x replicates. Have a sample table of data to illustrate. Note that this is sometimes now called a 'metapopulation design' because the spatial sampling is an EXPLICIT feature of the problem..., in contrast to classical population sampling, spatial sampling is not usually relevant (exception: distance sampling). (2) The common nature of the models is:

[Observations|latent state variable]

[latent state variable]

Latent state variable is always abundance or occurrence. The observation models have a lot in common in the sense that they have parameters to account for imperfect detection which is one of the unifying features of sampling biological populations.

Inference in **unmarked** is based on integrated likelihood wherein the latent state variable is marginalized out of the conditional likelihood.

Emphasize really strongly that a huge broad and diverse class of models is unified by these two features (the metapopulation sampling design and the hierarchical model structure).

1.3. Scope and features of unmarked

unmarked provides tools to assist researchers with every step of the analysis process, including data manipulation and exploration, model fitting, post-hoc analysis, model criticism, and model selection. Because of the multi-level structure of the data and models, covariate data can exist at both the state and detection level. To succinctly describe these data, **unmarked** uses a new data type called the unmarkedFrame (Section 3.1). **unmarked** provides functions that import data from various common formats and convert them into **unmarked**'s data types. Once imported, **unmarked** provides functions to summarize and subset these data in a manner familiar to users of R's more common data structures such as vectors, matrices, and `data.frame`.

unmarked provides a growing list of model-fitting functions designed for specific sampling methods. The fitting functions each find the maximum likelihood estimates of parameters from a particular model (Section 3.3) and return an object that can be easily manipulated. Methods exist for performing numerous post-hoc analyses such as requesting linear combinations of parameters, back-transforming parameters to constrained scales, determining confidence intervals, and evaluating goodness of fit. The model specification syntax of the fitting functions was designed to resemble the syntax of R's common fitting functions such as `lm` for fitting linear models.

Although there is existing software for fitting some of these models (Hines and MacKenzie 2002, e.g.,), there are a number of advantages to a unified framework within R. Many researchers are already familiar with R and use its powerful data manipulation and plotting capabilities. Sometimes many species are analyzed in tandem, so that a common method of aggregating and post-processing of results is needed, a task easily accomplished in R. Unlike other available software, **unmarked** makes it easy to map habitat-specific abundance and species distributions when combined with R's GIS capabilities. Another important advantage of **unmarked**'s approach is that researchers can easily simulate and analyze data within the same computational environment. This work flow permits simulation studies for power analysis calculations or the effectiveness of future sampling designs. All of this is made much simpler by analyzing the data within R, and using a single environment to complete all phases of the analysis is much less error-prone than switching between applications.

In this paper, Section 2 gives a brief summary of many of the models **unmarked** is capable of fitting. Section 3 describes general **unmarked** usage aided by a running data example.

2. Models implemented in unmarked

The list of models implemented in **unmarked** continues to grow as new models are developed. Table 1 shows the models available as of version 0.9-0. Rather than describe each fitting function in detail, this section provides a summary of several of the most common sampling techniques and how **unmarked** can be used to model the resulting data.

Model	**Fitting function**	**Data**	**Citation**
Single-season Occupancy	`occu`	unmarkedFrameOccu	(MacKenzie *et al.* 2002)
Abundance from presence-absence	`occuRN`	unmarkedFrameOccu	(Royle and Nichols 2003)
Abundance from repeated counts	`pcount`	unmarkedFramePCount	(Royle 2004b)
Distance-sampling	`distsamp`	unmarkedFrameDS	(Royle *et al.* 2004)
Multinomial Counts	`multinomPois`	unmarkedFrameMPois	(Royle 2004a)
Repeated Multinomial Counts	`gmultmix`	unmarkedFrameGMM	(Royle 2004a)
Multi-season Occupancy	`colext`	unmarkedMultFrame	(MacKenzie *et al.* 2003)
Multi-season Abundance	`pcountOpen`	unmarkedPCO	(Dail and Madsen 2011)

Table 1: Models handled by unmarked along with their associated fitting functions (Section 2) and data type (Section 3.1).

2.1. Single-season Occurrence data

An important state variable in ecological research is species occurrence (or 'site occupancy'), say Z_i, a binary state variable such that $Z_i = 1$ if site i is occupied by a species and $Z_i = 0$ otherwise. Much interest is focused on estimating functions of species occurrence (e.g., number of occupied sites) or to identify factors that are associated with the changes in the probability of a site being occupied, i.e., $\psi_i = \Pr(Z_i = 1)$. To estimate these parameters, researchers employ a sampling design, whereby surveyors visit a sample of M sites and record the binary response Y_{ij} of species detection ($Y = 1$) or non-detection ($Y = 0$) during $j = 1, \ldots, J_i$ visits to the ith site during a 'season' (MacKenzie *et al.* 2002). A key feature of species occurrence surveys is that false absences are possible, so that a species might go undetected ($Y_{ij} = 0$) even if it is present ($Z_i = 1$). A set of parameters p_{ij} accounts for this observation process, where p_{ij} is the probability of detecting the species at site i during the jth survey occasion given that site i is truly occupied. That is, $p_{ij} = \Pr(Y_{ij} = 1|Z_i = 1)$. The key assumptions made when modelling these data are that the occupancy state at a site remains constant throughout the season and repeated visits at a site are independent. The first assumption is commonly referred to as the 'population closure' assumption, meaning that no births, deaths, movements on or off the site are believed to occur during the season. Season, therefore, will generally refer to a very short time frame, such as a few months during a breeding season, or even a few minutes if repeated visits are made in quick succession. Replicate samples (i.e., $J > 1$) are necessary to obtain information about the detection rate separate from the occupancy rate.

The following hierarchical model describes the joint distribution of the observations conditional on the latent occupancy state, and the marginal distribution of the latent occupancy state variable:

$$Z_i \sim \text{Bernoulli}(\psi_i) \quad \text{for } i = 1, 2, \ldots, M \tag{1}$$

$$Y_{ij}|Z_i \sim \text{Bernoulli}(Z_i p_{ij}) \quad \text{for } j = 1, 2, \ldots, J_i \tag{2}$$

The likelihood can be obtained by removing the latent Z variables by marginalization, yielding the likelihood:

$$L(Y_{ij}|p, \psi) = \prod_i^M \left\{ \prod_j^J \left(p^{Y_{ij}} (1-p)^{1-Y_{ij}} \right) \psi + I(Y_{i.} = 0)(1-\psi) \right\}. \tag{3}$$

The model and likelihood are easily generalizable to accommodate covariates on detection probability or occupancy. Variables that are related to the occupancy state are modeled as

$$\text{logit}(\psi_i) = \mathbf{x}_i' \beta, \tag{4}$$

where $\mathbf{x}_i$ is a vector of site-level covariates and β is a vector of their corresponding effect parameters. Similarly, the probability of detection can be modeled with

$$\text{logit}(p_{ij}) = \mathbf{v}_{ij}' \alpha, \tag{5}$$

where $\mathbf{v}_{ij}$ is a vector of observation-level covariates and α is a vector of their corresponding effect parameters. Examples of site-level covariates include habitat characteristics such as vegetation height. Observation-level covariates could include time of day, date, wind speed, or other factors that might affect detection probability.

The likelihood for the single-season occupancy model, as maximized by the `occu` function, is

A generalization of this model that relaxes the population closure assumption is fit by the `colext` function, described subsequently.

2.2. Repeated count data

Occupancy is a crude summary of population structure and dynamics and, in practice, considerable effort is focused on obtaining information about abundance. A common sampling design for obtaining abundance information, analogous to the basic occupancy design described above, is to repeatedly visit a sample of M sites and record the number of unique individuals observed at each site. Similar assumptions are made as with occurrence data: (1) abundance at a site remains constant during a season and (2) counts at a site are independent. Royle (2004b) presented the following hierarchical model for repeated count data. Let N_i be the unobserved total number of individuals using a site and define Y_{ij} as the number of individuals observed during the jth visit. Then,

$$N_i \sim f(\lambda_i, \theta) \quad \text{for } i = 1, 2, \ldots, M \tag{6}$$

$$Y_{ij}|N_i \sim \text{Binomial}(N_i, p_{ij}) \quad \text{for } j = 1, 2, \ldots, J_i, \tag{7}$$

where λ_i is the abundance rate at site i and p_{ij} is the detection probability during the jth visit to site i. f is a discrete distribution with support restricted to $N_i \geq 0$ and θ are extra parameters of f other than the location parameter, λ_i. **unmarked** currently supports f as Poisson or negative binomial. In the Poisson case, there is no θ. In the negative binomial case, θ is a dispersion parameter, which is useful when overdispersion is suspected.

As with the occupancy model, covariates may be included at either the state (here, abundance) or detection levels, but abundance is modeled through a log link to enforce it's positivity constraint.

$$\log(\lambda_i) = \mathbf{x}_i'\beta, \tag{8}$$

where $\mathbf{x}_i$ is a vector of site-level covariates and β is a vector of their corresponding effect parameters. Similarly, the probability of detection can be modeled with

$$\text{logit}(p_{ij}) = \mathbf{v}_{ij}'\alpha, \tag{9}$$

where $\mathbf{v}_{ij}$ is a vector of observation-level covariates and α is a vector of their corresponding effect parameters.

The N_i variables are latent and so analysis is easily carried out based on the integrated likelihood obtained by marginalizing each N_i from the conditional likelihood Royle *et al.* (2004):

$$L(Y_{ij}|p,\lambda) = \prod_i^M \left\{ \sum_{N_i=max(\mathbf{Y_i})}^{\infty} \left(\prod_j^J \frac{N_{i!}}{(N_i - Y_{ij})!} p^{Y_{ij}} (1-p)^{N_i - Y_{ij}} \right) \frac{e^{-\lambda}\lambda^N}{N!} \right\}. \tag{10}$$

This likelihood can be maximized using the `pcount` function.

Two generalizations of this model can also be fit in **unmarked** by the `gmultmix` and `pcountOpen` functions. As with `colext`, these functions relax the population closure assumption and allow for the modelling of population dynamics.

Note that, because we are using classical methods as opposed to Bayesian methods, the latent variable `N` must be integrated out of the likelihood, and the user must choose a finite range for the discrete integration. The upper limit of this range can be specified using the `K` argument. It should be set sufficiently higher than the maximum observed count so that the MLEs are not affected. However, users should be aware that compuatation time increases with `K`.

2.3. General multinomial-Poisson mixture model

Here we discuss a more general class of models that can be modified to fit a variety of sampling methods, the multinomial-Poisson model (Royle 2004a). The general form of this model is

$$N_i \sim \text{Poisson}(\lambda_i) \quad \text{for } i = 1, 2, \ldots, M \tag{11}$$

$$\left(\begin{matrix} \mathbf{Y}_i \\ N_i - \sum_{j=1}^{J} Y_{ij} \end{matrix} \right) \Bigg| N_i \sim \text{Multinomial}\left(N_i, \begin{pmatrix} \boldsymbol{\pi}_i \\ \pi_i^* \end{pmatrix} \right) \tag{12}$$

where N_i is the latent abundance at site i as with the repeated count model, and $\boldsymbol{\pi}_i = (\pi_{i1}, \pi_{i2}, \ldots, \pi_{iJ})'$ is the vector of cell probabilities of observing responses in the J possible categories, and $\mathbf{Y}_i$ is the J-vector of counts that were actually observed. In general, $\boldsymbol{\pi}_i$ is determined by the specific sampling method and $\sum_j \pi_{ij} \leq 1$ because detection is imperfect and $\pi_i^* = 1 - \sum_j \pi_{ij}$ is the probability of the species escaping detection at site i.

It is worth noting here that the replication here is of a different nature – not over time necessarily – but still effectively replication as far as the design goes. That is you have a 'sites by K' matrix of counts its just that, with a multinomial protocol., the replicate counts are DEPENDENT instead of independent - and this is modeled with the multinomial observation model. Not sure where to say this blurb but probably should be in the first paragraph of this section.

To illustrate the likelihood under a multinomial observation model, suppose that a sampling method produces a multinomial observation with 3 observable frequencies at each site. As before, N_i are latent variables, and so inference is based on the integrated likelihood of $\mathbf{y}_i$ which is:

$$L(\mathbf{Y}_i|\mathbf{p}, \lambda) = \prod_{i=1}^{N} \left\{ \sum_{M_i=0}^{\infty} \left(\frac{N_i!}{y_{i1}! y_{i2}! y_{i3}! y_{i0}!} \pi_1^{y_{i1}} \pi_2^{y_{i2}} \pi_3^{y_{i3}} \pi^{*N_i - y_{i.}} \right) \frac{e^{-\lambda} \lambda^N}{N!} \right\} \tag{13}$$

For the specific case of a multinomial/Poisson mixture, the likelihood simplifies analytically (to a product-Poisson likelihood). The function `multinomPois` can be used to maximize this likelihood.

For multinomial-Poisson sampling methods, the actual observations are an underlying categorical detection variable with $M \leq J$ levels so that the J-dimensional $\mathbf{Y}_i$ is derived from the M-dimensional raw counts in some sampling method-specific manner. Thus, it is necessary to model the detection at the raw observation level, denoted p_{ik} for $k = 1, 2, \ldots, M$ at site i. Then we derive the multinomial cell probabilities $\boldsymbol{\pi}_i$ through the sampling technique-specific relationship $\pi_{ij} = g(p_{ik})$ where p_{ik} is the underlying probability of detection and g is some sampling method-specific function.

Thus, the only two requirements to adapt **unmarked**'s general multinomial Poisson modeling to a new sampling method is to specify g and a binary 0-1 matrix that describes the mapping of elements of $\mathbf{p}_i = (p_{i1}, \dots, p_{iR})'$ to elements of $\boldsymbol{\pi}_i$. This mapping matrix, referred to in **unmarked** as `obsToY`, is necessary to consistently clean missing values from the data and relate observation-level covariates with the responses. The (j,k)th element of `obsToY` is 1 only if p_{ik} is involved in the computation of π_{ij}. The detection function g is called `piFun` in **unmarked**.

Covariates may be included in either the state (here, abundance) or detection models, through $\mathbf{p}_i$ (not $\boldsymbol{\pi}_i$).

$$\log(\lambda_i) = \mathbf{x}_i'\beta, \tag{14}$$

where $\mathbf{x}_i$ is a vector of site-level covariates and β is a vector of their corresponding effect parameters. Similarly, the probability of detection can be modeled with

$$\text{logit}(p_{ij}) = \mathbf{v}_{ij}'\alpha, \tag{15}$$

where $\mathbf{v}_{ij}$ is a vector of observation-level covariates and α is a vector of their corresponding effect parameters.

We now describe two common sampling methods that can be modeled with the multinomial-Poisson model: removal sampling and double observer sampling. These two methods are included in **unmarked**, but additional methods may easily be specified by defining a user-specified `piFun` function and `obsToY` matrix.

Removal sampling

Popular in fisheries, removal sampling is commonly implemented by visiting a sample of M sites J times each and trapping or otherwise capturing and then removing individuals at each visit. Thus, Y_{ij} is the number of individuals captured at the jth visit for $j = 1, 2, \dots, J$.

We can specify g for removal sampling as follows. The probability of an individual at site i being captured on the first visit is $\pi_{i1} = p_{i1}$. The probability of capture on the jth visit is

$$\pi_{ij} = \prod_{k=1}^{j-1}(1 - p_{ik})p_{ij}, \tag{16}$$

for $j = 2, \dots, J$ and the probability of not being sampled is

$$\pi_{i,J+1} = \prod_{j=1}^{J}(1 - p_{ij}) \tag{17}$$

Or, equivalently,

$$\pi_i = \begin{pmatrix} p_{i1} \\ (1 - p_{i1})p_{i2} \\ \vdots \\ \prod_{j=1}^{J}(1 - p_{ij})p_{iJ} \end{pmatrix} \tag{18}$$

Thus, the mapping matrix is an $J \times J$ matrix with ones in the upper triangle,

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & 1 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \tag{19}$$

Double observer sampling

Double observer sampling involves collecting data by a team of two surveyors simultaneously visiting a site. Each observer records a list of detected animals and at the end of the survey, the two observers attempt to reconcile their counts. If individuals are not uniquely marked, this may be a difficult task in practice; however, assuming that individuals can be distinguised, the data at each site are a vector of length three $\mathbf{Y}_i$, corresponding to the numbers of individuals seen by only observer one, only observer two, and both observers. Thus, for double observer sampling, g is defined as follows.

$$\pi_i = \begin{pmatrix} p_{i1}(1-p_{i2}) \\ (1-p_{i1})p_{i2} \\ p_{i1}p_{i2} \\ (1-p_{i1})(1-p_{i2}). \end{pmatrix} \tag{20}$$

The `obsToY` mapping matrix for double observer sampling is the following 3×2 matrix.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \tag{21}$$

2.4. Multi-season occupancy data

Sometimes the study objective is to understand the dynamics of the occupancy state over time. To obtain such information researchers conduct repeated occupancy studies (see Section 2.1) at the same sample of sites over consecutive seasons (MacKenzie *et al.* 2003) and seek to estimate probabilities of colonization (γ_{it}) and extinction (ϵ_{it}), where colonization is the change of an unoccupied site to occupied and extinction if the change of an occupied site to unoccupied. If the occupancy status is assumed to evolve according to a Markov process, then a 2-state finite hidden Markov model describes these data. Let Y_{itj} denote the observed species occurrence status at visit j during season t to site i. Then

$$Z_{i1} \sim \text{Bernoulli}(\psi) \tag{22}$$

$$Z_{it} \sim \begin{cases} \text{Bernoulli}(\gamma_{i(t-1)}) & \text{if } Z_{i(t-1)} = 0 \\ \text{Bernoulli}(1-\epsilon_{i(t-1)}) & \text{if } Z_{i(t-1)} = 1 \end{cases}, \tag{23}$$

$$\text{for } t = 2, 3, \dots, T$$

$$Y_{itj}|Z_{it} \sim \text{Bernoulli}(Z_{it}p_{itj}) \tag{24}$$

To define the likelihood of this model, let $\phi_0 = (\psi, 1-\psi)$ and

$$\theta_{\mathbf{Y_{it}}} = \begin{pmatrix} \prod_j^J pI(Y_{ijt}=0)(1-p_{ijt}) \\ I(\sum_j^J Y_{ijt}=0) \end{pmatrix} \tag{25}$$

when I(*arg*) is the indicator function returning 1 if *arg* is satisfied and 0 otherwise. Furthermore, let the matrix of one-step Markov transition probabilities be

$$\Phi_t = \begin{pmatrix} 1-\epsilon & \epsilon \\ 1-\gamma & \gamma \end{pmatrix} \tag{26}$$

The likelihood is then

$$L(Y_{ijt}|p, \phi, \epsilon, \gamma) = \prod_i^R \left\{ \phi_0 \left\{ \prod_t^{T-1} \Phi_t \theta_{Y_{it}} \right\} \theta_{Y_{iT}} \right\}, \tag{27}$$

which is maximized by the `colext` function.

3. unmarked usage

unmarked provides data structures, fitting syntax, and post-processing that form a cohesive framework for site-based ecological data analysis. In order to achieve these goals, **unmarked** uses the S4 class system (Chambers 2008). As R's most modern system of class-based programming, S4 allows customization of functions, referred to as methods, to specific object classes and superclasses. For example, when the generic `predict` method is called with any **unmarked** model fit object as an argument, the actual `predict` implementation depends on the specific model that was fit. Use of class-based programming can provide more reliable and maintainable software while also making the program more user-friendly (Chambers 2008).

3.1. Preparing data

unmarked uses a custom S4 data structure called the `unmarkedFrame` to store all data and metadata related to a sampling survey. Although this at first appears to add an extra layer of work for the user, there are several reasons for this design choice. The multilevel structure of the models means that standard rectangular data structures such as `data.frames` or matrices are not suitable for storing the data. For example, covariates might have been measured separately at the site level and at the visit level. Furthermore, the length of the response vector $\mathbf{Y}_i$ at site i might differ from the number of observations at the site as in the multinomial Poisson model. In some cases, metadata of arbitrary dimensions may need to be associated with the data. For example, in distance sampling it is necessary to store the units of measure and the survey design type. Aside from these technical reasons, Gentleman (2009) pointed out that the use of such portable custom data objects can simplify future reference to previous analyses, an often neglected aspect of research. Repeated fitting calls using the same set of data require less code repetition if all data is contained in a single object. Finally, calls to fitting functions have a cleaner appearance with a more obvious purpose when the call is not buried in data arguments. The parent data class is called an unmarkedFrame and each **unmarked** fitting function has its own data type that extends the unmarkedFrame. Thus, the first step when using **unmarked** is to import data into the proper type of unmarkedFrame. To ease this step, **unmarked** includes several helper functions to automatically convert data into an unmarkedFrame: `csvToUMF` which imports data directly from a comma-separated value text file, `formatWide` and `formatLong` which convert data from data frames, and the family of unmarkedFrame constructor functions.

An `unmarkedFrame` object contains components referred to as slots, which hold the data and metadata. All `unmarkedFrame` objects contain a slot for the observation matrix `y`, a `data.frame` of site-level covariates `siteCovs`, and a `data.frame` of observation-level covariates `obsCovs`. The `y` matrix is the only required unmarkedFrame slot. Each row of `y` contains either the observed counts or presence-absence data at each of the R sites. `siteCovs` is an M-row `data.frame` with a column for each site-level covariate. `obsCovs` is an MJ-row `data.frame` with a column for each observation-level covariate. Thus each row of `obsCovs` corresponds to a particular observation, with the order corresponding to site varying slower and observation within site varying faster. Both `siteCovs` and `obsCovs` can contain `NA` values corresponding to unbalanced or missing data. If any terms in the model specification are missing for that observation, **unmarked** automatically removes observations. **unmarked** provides constructor functions to make creating unmarkedFrames straightforward. For each specific data type, specific types of unmarkedFrames extend the basic unmarkedFrame to handle model-specific nuances.

Importing repeated count data

Here is an example of creating an unmarkedFrame for repeated count data (Section 2.2). First, load the included data set of Mallard (*Anas platyrhynchos*) point counts from Kéry *et al.* (2005).

```
> library(unmarked)
> data(mallard)
```

Loading the mallard data makes three objects available within the R workspace. The matrix `mallard.y` contains the number of mallards counted at each of $M = 239$ sites on $J = 3$ visits. It is formatted such that each row is a site and each columnn corresponds to a visit. The site-level covariates are column vectors in the `mallard.site data.frame`, which has M rows. The observation-level covariates are a list object with separate $M \times J$ matrices for each observation-level covariate. The unmarkedFrame constructors can accept `obsCovs` in this list format or as a `data.frame` in the format described above.

The following call to `unmarkedFramePCount` organizes the observations and covariates into an object that can be passed to the data argument of the fitting function `pcount`.

```
> mallardUMF <- unmarkedFramePCount(y = mallard.y, siteCovs = mallard.site,
+     obsCovs = mallard.obs)
```

Printing unmarkedFrames shows them as `data.frames` to make it easier to verify that the data were converted correctly. Here we show the first five rows only.

```
> head(mallardUMF, 5)
```

```
Data frame representation of unmarkedFrame object.
  y.1 y.2 y.3   elev length forest ivel.1 ivel.2 ivel.3 date.1 date.2
1   0   0   0 -1.173  0.801 -1.156 -0.506 -0.506 -0.506 -1.761  0.310
2   0   0   0 -1.127  0.115 -0.501 -0.934 -0.991 -1.162 -2.904 -1.047
3   3   2   1 -0.198 -0.479 -0.101 -1.136 -1.339 -1.610 -1.690 -0.476
4   0   0   0 -0.105  0.315  0.008 -0.819 -0.927 -1.197 -2.190 -0.690
5   3   0   3 -1.034 -1.102 -1.193  0.638  0.880  1.042 -1.833  0.167
  date.3
1  1.381
2  0.596
3  1.453
4  1.239
5  1.381
```

The site-level covariates are elevation (elev), transect length (length), and the proportion of forest covering the site (forest). The two observation-level covariates are a measure of survey effort (ivel) and the date of the survey (date). Note that these variables are standardized to mean of 0 and unit variance. We can also check data contents with a quick summary.

```
> summary(mallardUMF)

unmarkedFrame Object

239 sites
Maximum number of observations per site: 3
Mean number of observations per site: 2.76
Sites with at least one detection: 40

Tabulation of y observations:
   0    1    2    3    4    7   10   12 <NA>
 576   54   11    9    6    1    1    1   58

Site-level covariates:
      elev                length                forest
 Min.   :-1.436e+00   Min.   :-4.945e+00   Min.   :-1.265e+00
 1st Qu.:-9.565e-01   1st Qu.:-5.630e-01   1st Qu.:-9.560e-01
 Median :-1.980e-01   Median : 4.500e-02   Median :-6.500e-02
 Mean   :-4.603e-05   Mean   :-2.929e-05   Mean   : 6.695e-05
 3rd Qu.: 9.940e-01   3rd Qu.: 6.260e-01   3rd Qu.: 7.900e-01
 Max.   : 2.434e+00   Max.   : 2.255e+00   Max.   : 2.299e+00

Observation-level covariates:
      ivel                 date
 Min.   :-1.753e+00   Min.   :-2.904e+00
 1st Qu.:-6.660e-01   1st Qu.:-1.119e+00
 Median :-1.390e-01   Median :-1.190e-01
 Mean   : 1.504e-05   Mean   : 7.259e-05
 3rd Qu.: 5.490e-01   3rd Qu.: 1.310e+00
 Max.   : 5.980e+00   Max.   : 3.810e+00
 NA's   : 5.200e+01   NA's   : 4.200e+01
```

The summary reveals that only 40 sites have at least one detection. The tabulation of y observations provides additional evidence of sparse counts, with no mallards being detected during 576 of the surveys. The 58 NA values correspond to missing data.

Importing removal sampling data

To illustrate the slightly different syntax for removal sampling data example, we will import data from a removal survey of ovenbirds (*Seiurus aurocapillus*) described by Royle (2004a). The data consist of a list named `ovendata.list` with a matrix `data` containing the removal counts for 4 visits and a `data.frame` called `covariates` containing site-level covariates information.

```
> data(ovendata)
> head(ovendata.list$data, 10)
```

```
      [,1] [,2] [,3] [,4]
 [1,]    0    0    0    0
 [2,]    1    0    0    0
 [3,]    0    0    0    0
 [4,]    0    0    0    0
 [5,]    0    0    0    0
 [6,]    0    0    0    0
 [7,]    0    0    0    0
 [8,]    0    0    0    0
 [9,]    2    0    0    0
[10,]    1    0    0    0
```

Each row of the response matrix corresponds to a site, and each column is a removal occasion. Thus, it is evident that ovenbirds were detected at three of the first ten sites and no new birds were detected after the first removal occasion. The only additional specification required when importing removal data is to specify the particular type of multinomial-Poisson data as `removal` via the `type` argument.

```
> ovenFrame <- unmarkedFrameMPois(ovendata.list$data, siteCovs = ovendata.list$covariates,
+     type = "removal")
> summary(ovenFrame)
```

```
unmarkedFrame Object

70 sites
Maximum number of observations per site: 4
Mean number of observations per site: 4
Sites with at least one detection: 44
```

```
Tabulation of y observations:
   0    1    2    3 <NA>
 218   49   11    2    0

Site-level covariates:
      site          ufp              trba
 CAT003 : 1   Min.   : 1.17   Min.   : 50.0
 CAT004 : 1   1st Qu.: 8.20   1st Qu.: 85.0
 CAT011 : 1   Median :12.89   Median :100.0
 CAT013 : 1   Mean   :15.33   Mean   :103.4
 CAT017 : 1   3rd Qu.:24.80   3rd Qu.:122.5
 CAT023 : 1   Max.   :37.89   Max.   :180.0
 (Other):64
```

Preparing multi-season occupancy data

The data structures are more complex when surveys occurred during more than one season. In this case, the covariates can occur at the site-, season-, and observation-levels. The following toy examples demonstrates how to prepare data for the `colext` fitting function if there were only four sites, two visits per season, and three seasons.

```
> M <- 4
> J <- 2
> T <- 3
> y <- matrix(0:1, nrow = M, ncol = J * T)
> umf <- unmarkedMultFrame(y = y, numPrimary = T)
> summary(umf)
```

```
unmarkedFrame Object

4 sites
Maximum number of observations per site: 6
Mean number of observations per site: 6
Number of primary survey periods: 3
Number of secondary survey periods: 2
Sites with at least one detection: 2

Tabulation of y observations:
   0    1 <NA>
  12   12    0
```

The function `unmarkedMultFrame` accepts `siteCovs` and `obsCovs` like all other unmarked-Frame classes, and it also has an argument `yearlySiteCovs` that can accept a list of $M \times T$ `data.frame`s containing season-level covariates.

3.2. Manipulating unmarkedFrames

The various components of `unmarkedFrames` can be extracted and subsetted in a manner similar to the methods used to manipulate standard R objects. Subsetting can be accomplished using the 'bracket' notation. For example, the first five rows of data and the first two removal occasions can be extracted from the ovenbird removal study using

```
> ovenFrame[1:5, 1:2]

Data frame representation of unmarkedFrame object.
  y.1 y.2   site   ufp  trba
1   0   0 CAT003 23.43  62.5
2   1   0 CAT004 15.62  70.0
3   0   0 CAT011 35.54  90.0
4   0   0 CAT013 25.78  60.0
5   0   0 CAT017 25.00 122.5
```

In some cases, the covariate data may need to be manipulated after the `unmarkedFrame` has been created. The following code demonstrates how to extract the site-level covariates, standardize those that are continuous (columns 2 and 3), and reinsert them back into the `unmarkedFrame`.

```
> sc <- siteCovs(ovenFrame)
> sc[, 2:3] <- scale(sc[, 2:3])
> siteCovs(ovenFrame) <- sc
```

3.3. Fitting models

As introduced in Section 2, each class of models has a corresponding fitting function. For example, to fit a repeated count model, we call `pcount`. Table 1 provides a summary of all models that **unmarked** currently fits. With the exception of the 'open' population models (`colext`, `gmultmix`, and `pcountOpen`), all fitting functions use a double right-hand sided formula syntax that expresses the hierarchical model and data structures . Specifically, covariates affecting the detection process are specified following the first tilde, and covariates of the state process follow the second tilde. No left-hand side of the formula is specified because the unmarkedFrame defines the response variable uniquely as the `y` slot.

Fitting a repeated count model

Continuing the Mallard example, the following call to `pcount` fits a binomaial-Poisson mixture model (Section 2.2). The following code specifies that detection probability p should be modeled by day of year, including a quadratic term. We also wish to model abundance using elevation and proportion of area forested. As described in Section 2.2, covariates of detection are on the logit-scale and covariates of abundance are on the log scale for the repeated count model.

```
> fm.mallard.1 <- pcount(~date + I(date^2) ~ elev + forest,
+     data = mallardUMF)
> fm.mallard.1
```

```
Call:
pcount(formula = ~date + I(date^2) ~ elev + forest, data = mallardUMF)

Abundance:
            Estimate    SE     z  P(>|z|)
(Intercept)   -1.853 0.236 -7.87 3.62e-15
elev          -1.232 0.234 -5.27 1.36e-07
forest        -0.756 0.165 -4.60 4.32e-06

Detection:
            Estimate     SE       z P(>|z|)
(Intercept)   0.3071 0.2149  1.4290 0.15301
date         -0.4091 0.1482 -2.7612 0.00576
I(date^2)    -0.0077 0.0853 -0.0903 0.92801

AIC: 521.2001
```

This initial fit suggests that Mallard abundance decreases with increasing elevation and forest. It also looks like a linear model might suffice for the detection model, so we subsequently fit the linear detection model as follows:

```
> fm.mallard.2 <- pcount(~date ~ elev + forest, data = mallardUMF)
> fm.mallard.2

Call:
pcount(formula = ~date ~ elev + forest, data = mallardUMF)

Abundance:
            Estimate    SE     z  P(>|z|)
(Intercept)   -1.857 0.232 -7.99 1.37e-15
elev          -1.236 0.230 -5.38 7.59e-08
forest        -0.756 0.164 -4.60 4.23e-06

Detection:
            Estimate    SE     z  P(>|z|)
(Intercept)    0.298 0.188  1.58 0.113983
date          -0.401 0.114 -3.51 0.000449

AIC: 519.2081
```

This seems to be a better model according to both the Wald p-value and AIC. The result suggests that detection probability decreases during the course of a year.

Fitting a multinomial-Poisson model

Here we demonstrate fitting a multinomial-Poisson mixture model to removal sampling data. The Ovenbird data has no observation-level covariates, so detection probability is assumed constant across visits. It is not necessary to specify that removal sampling was used when fitting the model because this information is already stored in the `ovenFrame` data. We model abundance as a function of understory forest coverage (`ufp`) and average basal tree area (`trba`).

```
> fm.oven.1 <- multinomPois(~1 ~ ufp + trba, ovenFrame)
> fm.oven.1

Call:
multinomPois(formula = ~1 ~ ufp + trba, data = ovenFrame)

Abundance:
            Estimate    SE      z P(>|z|)
(Intercept)    0.102 0.119  0.864   0.388
ufp            0.100 0.126  0.794   0.427
trba          -0.171 0.135 -1.262   0.207

Detection:
 Estimate    SE    z P(>|z|)
    0.288 0.233 1.24   0.217

AIC: 326.1387
```

3.4. Examining model fits

Objects returned by **unmarked**'s fitting functions also make use of the S4 class system. All model fit objects belong to the unmarkedFit parent class. Thus, common operations such as extracting coefficient estimates, covariance matrices for estimates, and confidence intervals have been adapted to behave similar to R's base functions.

For example, we can extract estimated coefficients either from the entire model, or from the state or detection levels by specifying the `type` argument.

```
> coef(fm.mallard.2)

   lam(Int)   lam(elev) lam(forest)      p(Int)     p(date)
 -1.8565048  -1.2355497  -0.7555178   0.2977558  -0.4006066

> coef(fm.mallard.2, type = "state")

   lam(Int)   lam(elev) lam(forest)
 -1.8565048  -1.2355497  -0.7555178
```

To check which types are available for a model, use the `names` method.

```
> names(fm.mallard.2)
```

```
[1] "state" "det"
```

Similarly, the `vcov` function extracts the covariance matrix of the estimates, using the observed Fisher information by default.

```
> vcov(fm.mallard.2)
```

```
                 lam(Int)    lam(elev)   lam(forest)       p(Int)
lam(Int)     0.054013257  0.040740133  0.0080777484 -0.006188545
lam(elev)    0.040740133  0.052810074 -0.0084624468 -0.001751201
lam(forest)  0.008077748 -0.008462447  0.0269780875  0.001779971
p(Int)      -0.006188545 -0.001751201  0.0017799714  0.035490236
p(date)     -0.001785422 -0.003393574  0.0005126907  0.003448530
                  p(date)
lam(Int)    -0.0017854217
lam(elev)   -0.0033935742
lam(forest)  0.0005126907
p(Int)       0.0034485301
p(date)      0.0130286462
```

`vcov` also accepts a `type` argument, as does the convenience method `SE`, which returns standard errors from the square root of the diagonal of the covariance matrix.

```
> SE(fm.mallard.2, type = "state")
```

```
   lam(Int)   lam(elev) lam(forest)
  0.2324075   0.2298044   0.1642501
```

Nonparametric bootstrapping can also be used to estimate the covariance matrix. **unmarked** implements a two-stage bootstrap in which the sites are first drawn with replacement, and then within each site, the observations are drawn with replacement. First, bootstrap draws must be taken using the `nonparboot`. `nonparboot` returns a new version of the unmarkedFit object with additional bootstrap sampling information. Thus, this new fit must be stored, either in a new fit object or the same one, and then subsequently queried for bootstrap summaries. In the following, bootstrapping can be quite slow. For these examples, we illustrate with the removal sampling data simply because computations are much faster. However, bootstrapping is available for any of the models that **unmarked** fits.

```
> set.seed(1234)
> fm.oven.1 <- nonparboot(fm.oven.1, B = 100)
```

```
> SE(fm.oven.1, type = "state")
```

```
 lambda(Int)  lambda(ufp) lambda(trba)
   0.1185431    0.1262615    0.1353444
```

```
> SE(fm.oven.1, type = "state", method = "nonparboot")

 lambda(Int)  lambda(ufp) lambda(trba)
   0.1256949    0.1156153    0.1197142
```

It looks like bootstrapping and asymptotic standard errors are close. The summary now states the number of bootstrap samples.

```
> summary(fm.oven.1)

Call:
multinomPois(formula = ~1 ~ ufp + trba, data = ovenFrame)

Abundance (log-scale):
            Estimate    SE      z P(>|z|)
(Intercept)    0.102 0.119  0.864   0.388
ufp            0.100 0.126  0.794   0.427
trba          -0.171 0.135 -1.262   0.207

Detection (logit-scale):
 Estimate    SE    z P(>|z|)
    0.288 0.233 1.24   0.217

AIC: 326.1387
Number of sites: 70
optim convergence code: 0
optim iterations: 33
Bootstrap iterations: 100
```

Additional bootstrap samples can be drawn by calling `nonparboot` again.

```
> fm.oven.1 <- nonparboot(fm.oven.1, B = 100)
```

Confidence intervals can be requested for the coefficients at either stage of the model. By default, the asymptotic normal approximation is used.

```
> confint(fm.oven.1, type = "state", level = 0.95)

                  0.025      0.975
lambda(Int)  -0.1299722 0.33470834
lambda(ufp)  -0.1471741 0.34776195
lambda(trba) -0.4361311 0.09440937
```

Profile confidence intervals are also available upon request. This can take some time, however, because for each parameter, a nested optimization within a root-finding algorithm is being used to find the profile limit.

```
> ci <- confint(fm.oven.1, type = "state", level = 0.95,
+     method = "profile")

Profiling parameter 1 of 3 ... done.
Profiling parameter 2 of 3 ... done.
Profiling parameter 3 of 3 ... done.

> ci

                  0.025      0.975
lambda(Int)  -0.1390786 0.32676614
lambda(ufp)  -0.1477724 0.34811770
lambda(trba) -0.4368444 0.09469605
```

The profile confidence intervals and normal approximations are quite similar here.

Linear combinations of estimates

Often, meaningful hypotheses can be addressed by estimating linear combinations of coefficient estimates. Linear combinations of coefficient estimates can be requested with `linearComb`. Continuing the Ovenbird example, the following code estimates the log-abundance rate for a site with `ufp = 0.5` and `trba = 0`.

```
> (lc <- linearComb(fm.oven.1, type = "state", coefficients = c(1,
+     0.5, 0)))

Linear combination(s) of Abundance estimate(s)

 Estimate    SE (Intercept) ufp trba
    0.153 0.130           1 0.5    0
```

Multiple sets of coefficients may be supplied as a matrix. The following code requests the estimated log-abundance for sites with `ufp = 0.5` and `trba = 1`.

```
> (lc <- linearComb(fm.oven.1, type = "state", coefficients = matrix(c(1,
+     0.5, 0, 1, 1, 0), 2, 3, byrow = TRUE)))

Linear combination(s) of Abundance estimate(s)

  Estimate    SE (Intercept) ufp trba
1    0.153 0.130           1 0.5    0
2    0.203 0.166           1 1.0    0
```

Standard errors and confidence intervals are also available for linear combinations of parameters. By requesting nonparametric bootstrapped standard errors, **unmarked** uses the samples that were drawn earlier.

```
> SE(lc)

[1] 0.1296526 0.1659460

> SE(lc, method = "nonparboot")

[1] 0.1354092 0.1659877

> confint(lc)

       0.025     0.975
1 -0.1015993 0.4066294
2 -0.1225862 0.5279102
```

Back-transforming linear combinations of coefficients

Estimates of linear combination back-transformed to the native scale are likely to be more interesting than the direct linear combinations. For example, the logistic transformation is applied to estimates of detection rates, resulting in a probability bound between 0 and 1. This is accomplished with the `backTransform`. Standard errors of back-transformed estimates are estimated using the delta method. Confidence intervals are estimated by back-transforming the confidence interval of the original linear combination.

```
> (btlc <- backTransform(lc))

Backtransformed linear combination(s) of Abundance estimate(s)

  Estimate    SE LinComb (Intercept) ufp trba
1     1.16 0.151   0.153           1 0.5    0
2     1.22 0.203   0.203           1 1.0    0

Transformation: exp

> SE(btlc)

[1] 0.1510141 0.2032272

> SE(btlc, method = "nonparboot")

[1] 0.1577193 0.2032782

> confint(btlc)

      0.025    0.975
1 0.9033915 1.501747
2 0.8846296 1.695386
```

Model selection

unmarked performs AIC-based model selection for structured lists of unmarkedFit objects. To demonstrate, we fit a few more models to the Ovenbird removal data, including an interaction model, two models with single predictors, and a null model with no predictors.

```
> fm.oven.2 <- update(fm.oven.1, formula = ~1 ~ ufp * trba)
> fm.oven.3 <- update(fm.oven.1, formula = ~1 ~ ufp)
> fm.oven.4 <- update(fm.oven.1, formula = ~1 ~ trba)
> fm.oven.5 <- update(fm.oven.1, formula = ~1 ~ 1)
```

Now, we can organize the fitted models with the `fitList` function and the use the `modSel` method to rank the models by AIC.

```
> fmList <- fitList(Global = fm.oven.2, additive = fm.oven.1,
+     ufp = fm.oven.3, trba = fm.oven.4, Null = fm.oven.5)
> modSel(fmList)

         nPars    AIC delta AICwt cumltvWt
trba         3 324.77  0.00  0.35     0.35
ufp          3 325.73  0.96  0.21     0.56
additive     4 326.14  1.37  0.17     0.73
Null         2 326.28  1.51  0.16     0.90
Global       5 327.17  2.40  0.10     1.00
```

It looks like the best model includes only tree basal area as a predictor of abundance. We can examine this relationship using the `predict` method (Figure 1).

```
> preddata <- predict(fm3, type = "state", appendData = TRUE)
> library(ggplot2)
> qplot(trba, Predicted, data = preddata, geom = "line",
+     xlab = "Scaled Basal Tree Area", ylab = "Estimated Abundance") +
+     geom_ribbon(aes(x = trba, ymin = lower, ymax = upper),
+         alpha = 0.1) + theme_bw()
```

`predict` functions much like `linearComb` except that new data can be passed to it as a `data.frame` rather than a design matrix. When the first argument given to predict is a list of models created by `fitList`, `predict` computes model-averaged predictions, which may be useful in the presence of high model selection uncertainty.

Parametric bootstrap for goodness of fit

To conduct goodness of fit tests, **unmarked** provides a generic parametric bootstrapping function. It simulates data from the fitted model and applies a user-defined function that returns a fit-statistic such as the sum of squared residuals. Beyond serving as a tool to evaluate goodness of fit, `parboot` can be used to characterize uncertainty in any derived quantity of interest. Here we compute a chi-squared statistic to evaluate goodness of fit.

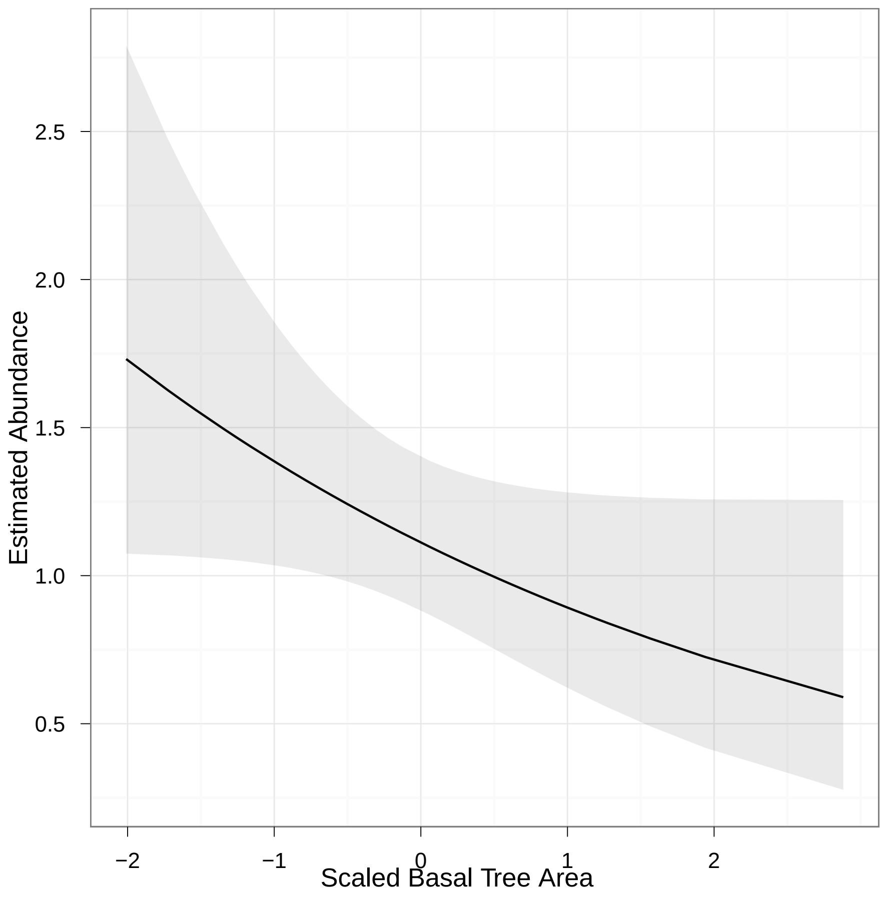

Figure 1: Examine estimated abundance for Ovenbird removal data. Band is 95% confidence interval.

```
> set.seed(1234)
> chisq <- function(fm) {
+     observed <- getY(fm@data)
+     expected <- fitted(fm)
+     sum((observed - expected)^2/expected)
+ }
> pb <- parboot(fm.oven.1, statistic = chisq, nsim = 20)
> pb

Call: parboot(object = fm.oven.1, statistic = chisq, nsim = 20)

Parametric Bootstrap Statistics:
   t0 mean(t0 - t_B) StdDev(t0 - t_B) Pr(t_B > t0)
1 301           24.5             26.7        0.238

t_B quantiles:
     0% 2.5% 25% 50% 75% 97.5% 100%
```

```
t*1 232  238 258 273 299   326  342

t0 = Original statistic compuated from data
t_B = Vector of bootstrap samples

> plot(pb, main = "")
```

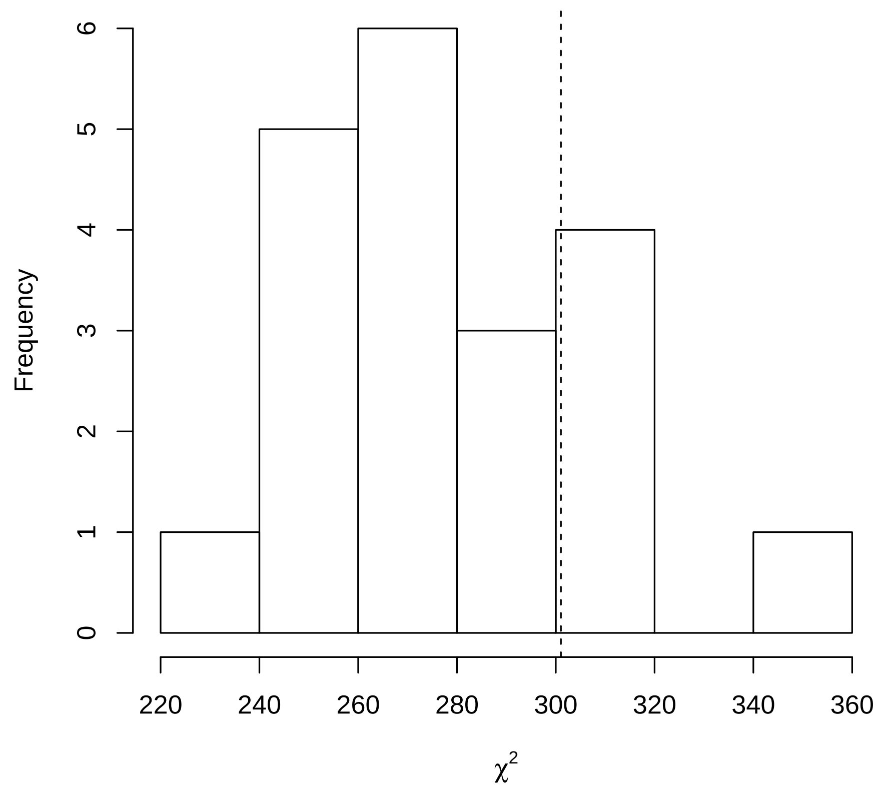

Figure 2: Graphically assess model fit by parametric bootstrapping. The dashed line is the observed chi-squared statistic. The histogram approximates the expected sampling distribution

The above call to `plot` with a parametric bootstrap object as the argument produces a useful graphic for assessing goodness of fit (Figure 2). The plot suggests that the model adequately explains these data.

4. Future directions for unmarked development

unmarked has become a stable and useful platform for the analysis of ecological data, but several areas of development could improve its utility. First, new models need to be added to cover the range of sampling techniques and population dynamics commonly encountered. Table 3 illustrates the current gaps that need to be filled. In most cases, models to fill these gaps have not been developed so more research is needed.

	Population Dynamics		
Sampling method	Closed	Open to movement	Open to demographic processes
Occurrence sampling	`occu`	–	`colext`
Repeated counts	`pcount`	–	`pcountOpen`
Removal sampling, double observer sampling, and other multinomial designs	`multinomPois`	`gmultmix`	–
Distance-sampling	`distsamp`	–	–

Table 2: Model fitting functions classified by sampling method and population dynamics.

Second, each of the models in **unmarked** assumes independence among sites. However, ecologists often use sampling methods such as cluster sampling that induce spatial dependence. Typically, this is done for logistical convenience, but because few methods are available to account for spatial correlation and imperfect detection probability, the spatial dependence is often ignored. Rather than this being a weakness of the sampling design, we envision that this dependence can be used as information regarding the spatial distribution of individuals.

Finally, Bayesian estimation could be implemented for many of these models. An important advantage of Bayesian analysis over classical methods is that the latent abundance or occurrence variables can be treated as formal parameters. Thus posterior distributions could easily be calculated for derived parameters such as the proportion of sites occupied. Bayesian analysis also would provide a natural framework for incorporating additional sources of random variation. For example, one could model heterogeneity among sites not accounted for by covariates alone.

Acknowledgements

The authors thank Andy Royle of the United States Geological Survey's Patuxent Wildlife Research Center, who provided initial funding and inspiration for this work.

References

Chambers JM (2008). *Software for data analysis: Programming with R*. Springer Verlag, New York, USA.

Dail D, Madsen L (2011). "Models for Estimating Abundance from Repeated Counts of an Open Metapopulation." *Biometrics*.

Gentleman R (2009). *R programming for bioinformatics*. Chapman & Hall/CRC, New York, USA.

Hines JE, MacKenzie DI (2002). "PRESENCE." URL `http://www.mbr-pwrc.usgs.gov/software/presence.html`.

Kéry M, Royle JA, Schmid H (2005). "Modeling avian abundance from replicated counts using binomial mixture models." *Ecological Applications*, **15**(4), 1450–1461. URL http://www.esajournals.org/doi/abs/10.1890/04-1120.

MacKenzie DI, Nichols JD, Hines JE, Knutson MG, Franklin AB (2003). "Estimating site occupancy, colonization, and local extinction when a species is detected imperfectly." *Ecology*, **84**(8), 2200–2207.

MacKenzie DI, Nichols JD, Lachman GB, Droege S, Royle JA, Langtimm CA (2002). "Estimating site occupancy rates when detection probabilities are less than one." *Ecology*, **83**(8), 2248–2255.

Royle JA (2004a). "Generalized estimators of avian abundance from count survey data." *Animal Biodiversity and Conservation*, **27**(1), 375–386.

Royle JA (2004b). "N-mixture models for estimating population size from spatially replicated counts." *Biometrics*, **60**(1), 108–115.

Royle JA, Dawson DK, Bates S (2004). "Modeling abundance effects in distance sampling." *Ecology*, **85**(6), 1591–1597.

Royle JA, Nichols JD (2003). "Estimating abundance from repeated presence-absence data or point counts." *Ecology*, **84**(3), 777–790.

Williams BK, Nichols JD, Conroy MJ (2002). *Analysis and Management of Animal Populations*. Academic Press, San Diego, USA.

Affiliation:

Ian Fiske
Department of Statistics
North Carolina State University
2311 Stinson Drive
Campus Box 8203
Raleigh, NC 27695-8203
E-mail: ijfiske@ncsu.edu

Richard Chandler
USGS Patuxent Wildlife Research Center
Gabrielson Lab, Room 226
12100 Beech Forest Rd.
Laurel, MD 20708
E-mail: rchandler@usgs.gov

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd