Journal of Statistical Software

October 2020, Volume 95, Issue 2. *doi: 10.18637/jss.v095.i02*

Estimating Animal Abundance with N-Mixture Models Using the R-INLA Package for R

Timothy D. Meehan
National
Audubon Society

Nicole L. Michel
National
Audubon Society

Håvard Rue
King Abdulla University
of Science and Technology

Abstract

Successful management of wildlife populations requires accurate estimates of abundance. Abundance estimates can be confounded by imperfect detection during wildlife surveys. N-mixture models enable quantification of detection probability and, under appropriate conditions, produce abundance estimates that are less biased. Here, we demonstrate how to use the R-**INLA** package for R to analyze N-mixture models, and compare performance of R-**INLA** to two other common approaches: **JAGS** (via the **runjags** package for R), which uses Markov chain Monte Carlo and allows Bayesian inference, and the **unmarked** package for R, which uses maximum likelihood and allows frequentist inference. We show that R-**INLA** is an attractive option for analyzing N-mixture models when (i) fast computing times are necessary (R-**INLA** is 10 times faster than **unmarked** and 500 times faster than **JAGS**), (ii) familiar model syntax and data format (relative to other R packages) is desired, (iii) survey-level covariates of detection are not essential, and (iv) Bayesian inference is preferred.

Keywords: abundance, detection, **JAGS**, N-mixture model, R, R-**INLA**, **unmarked**, wildlife.

1. Introduction

1.1. Background

Successful management of wildlife species requires accurate estimates of abundance (Yoccoz, Nichols, and Boulinier 2001). One common method for estimating animal abundance is direct counts (Pollock, Nichols, Simons, Farnsworth, Bailey, and Sauer 2002). Efforts to obtain accurate abundance estimates via direct counts can be hindered by the cryptic nature of many wildlife species, and by other factors such as observer expertise, weather, and habitat structure (Dénes, Silveira, and Beissinger 2015). The lack of perfect detection in wildlife surveys is common, and can cause abundance to be greatly underestimated (Wenger and Freeman 2008; Joseph, Elkin, Martin, and Possingham 2009).

In recent years, new survey designs and modeling approaches have enabled improved estimates of animal abundance that are less biased by imperfect detection (Dénes *et al.* 2015). One such survey design, termed a metapopulation design (Kéry and Royle 2010), involves repeat visits in rapid succession to each of multiple study sites in a study area. If, during repeat visits, the population is assumed to be closed (no immigration, emigration, reproduction or mortality; i.e., static abundance), then information on detections and non-detections during repeated counts can inform an estimate of detection probability. This detection probability can be used to correct abundance estimates for imperfect detection (Royle 2004).

Data resulting from this survey design are often modeled using an explicitly hierarchical statistical model referred to in the quantitative wildlife ecology literature as an N-mixture model (Royle and Nichols 2003; Dodd and Dorazio 2004; Royle 2004; Kéry, Royle, and Schmid 2005). One form of N-mixture model, a binomial mixture model, describes individual observed counts y at site i during survey j as coming from a binomial distribution with parameters for abundance N and detection probability p, where N per site is drawn from a Poisson distribution with an expected value λ. Specifically,

$$N_i \sim \text{Pois}(\lambda) \qquad \text{and} \qquad y_{i,j} \mid N_i \sim \text{Bin}(N_i, p). \tag{1}$$

The quantitiy λ is commonly modeled as a log-linear function of site covariates, as $\log(\lambda_i) = \beta_0 + \beta_1 x_i$. Similarly, p is commonly modeled as $\text{logit}(p_{i,j}) = \alpha_0 + \alpha_1 x_{i,j}$, a logit-linear function of site-survey covariates.

This estimation approach can be extended to cover K distinct breeding or wintering seasons, which correspond with distinct years for wildlife species that are resident during annual breeding or wintering stages (Kéry, Dorazio, Soldaat, Van Strien, Zuiderwijk, and Royle 2009). In this case, population closure is assumed across J surveys within year k, but is relaxed across years (Kéry *et al.* 2009). A simple specification of a multiple-year model is $N_{i,k} \sim \text{Pois}(\lambda_k)$, $y_{i,j,k} \mid N_{i,k} \sim \text{Bin}(N_{i,k}, p_k)$. Like the single-year specification, λ is commonly modeled using site and site-year covariates, and p using site-survey-year covariates.

There are other variations of N-mixture models that accommodate overdispersed Poisson counts through use of a negative binomial distribution (Kéry and Royle 2010), a zero-inflated Poisson distribution (Wenger and Freeman 2008), or survey-level random intercepts (Kéry and Schaub 2011), or underdispersed counts using mixtures of binomial and Conway-Maxwell-Poisson distributions (Wu, Holan, Nilon, and Wikle 2015). Yet other variations account for non-independent detection probabilities through use of a beta-binomial distribution (Martin, Royle, Mackenzie, Edwards, Kéry, and Gardner 2011), parse different components of detection through the use of unique covariates (O'Donnell, Thompson, and Semlitsch 2015), or relax assumptions of population closure (Chandler, Royle, and King 2011; Dail and Madsen 2011). We do not discuss all of these variations here, but refer interested readers to Dénes *et al.* (2015) for an overview, and to Barker, Schofield, Link, and Sauer (2018) for a discussion of assumptions and limitations.

The development of metapopulation designs and N-mixture models represents a significant advance in quantitative wildlife ecology. However, there are practical issues that sometimes act as barriers to adoption. Many of the examples of N-mixture models in the wildlife literature have employed Bayesian modeling software such as **WinBUGS** (Spiegelhalter, Thomas, Best, and Lunn 2003), **OpenBUGS** (Thomas 2014), **JAGS** (Plummer 2003), or Stan (Carpenter *et al.* 2017). These are extremely powerful and flexible platforms for analyzing hierarchical models, but they come with a few important challenges. First, many wildlife biologists are not accustomed to coding statistical models using the BUGS or Stan modeling syntax. While there are several outstanding resources aimed at teaching these skills (Royle and Dorazio 2008; Kéry 2010; Kéry and Schaub 2011; Kéry and Royle 2015; Korner-Nievergelt, Roth, von Felton, Guélat, Almasi, and Korner-Nievergelt 2015) learning them is, nonetheless, a considerable commitment. Second, while Markov chain Monte Carlo (MCMC) chains converge quickly for relatively simple N-mixture models, convergence for more complex models can take hours to days, or may not occur at all (Kéry and Schaub 2011).

There are other tools available for analyzing N-mixture models that alleviate some of these practical issues. The **unmarked** package (Fiske and Chandler 2011) for the R statistical computing software (R Core Team 2020) offers several options for analyzing N-mixture models within a maximum likelihood (ML) framework, with the capacity to accommodate overdispersed Poisson counts and dynamic populations. The model coding syntax used in **unmarked** is a simple extension of the standard R modeling syntax. Models are analyzed using ML, so model analysis is usually completed in a fraction of the time taken using MCMC. The familiar model syntax and rapid model evaluation of **unmarked** has undoubtedly contributed to the broader adoption of N-mixture models by wildlife biologists. However, it comes at a cost: loss of the intuitive inferential framework associated with Bayesian analysis.

Here we demonstrate analysis of N-mixture models using the R-**INLA** package (Rue, Riebler, Sørbye, Illian, Simpson, and Lindgren 2017) for R. The R-**INLA** package uses integrated nested Laplace approximation (INLA) to derive posterior distributions for a large class of Bayesian statistical models that can be formulated as latent Gaussian models (Rue, Martino, and Chopin 2009; Lindgren, Rue, and Lindstrom 2011). INLA was developed to allow estimation of posterior distributions in a fraction of the time taken by MCMC. Like **unmarked**, the model syntax used by the R-**INLA** package is a straightforward extension of the modeling syntax commonly used in R. Also, like **unmarked**, the computational cost of analyzing models with R-**INLA** is relatively low compared to MCMC. The R-**INLA** approach is different from **unmarked** in that inference about model parameters falls within a Bayesian framework.

1.2. Overall objectives

The purpose of this manuscript is to present a comparative analysis of N-mixture models that is centered on the R-**INLA** package. In the process, we employ both simulated and real count data sets, and analyze them using R-**INLA**, **JAGS**, via the **runjags** package (Denwood 2016) for R, and the **unmarked** package for R. In each case, we demonstrate how models are specified, how model estimates compare to simulation inputs and to each other, and how methods compare in terms of computational performance. When describing R-**INLA** analyses, we detail the format of input data and the content of analysis code, to facilitate readers conducting their own analyses.

We also explore a limitation of the R-**INLA** approach related to model specification. In particular, while it is possible to specify survey-level covariates for detection using **JAGS** and **unmarked**, this is not possible using R-**INLA**. Rather, survey-level covariates of detection must be averaged to the site or site-year level. Using an averaged detection covariate does allow accounting for site or site-year differences in survey conditions, should they occur.

However, in the process of averaging, information related to detection within a site or site-year combination is discarded, which could lead to biased detection and abundance estimates under certain conditions.

Much of the code used to conduct the R-**INLA** analyses is shown in the body of this manuscript. However, some repeated R-**INLA** code, code used in **JAGS** and **unmarked** analyses, and code related to generating figures, is not shown, for brevity. All code, fully commented, is available in the replication script `v95i02.R` available along with this paper. Regarding code, note that the R-**INLA** package is atypical among R packages in a few different ways. First, R-**INLA** is not available on the Comprehensive R Archive Network (CRAN), as are many other R packages. Second, R-**INLA** was initially called **INLA**, based on its origin as a stand-alone C program. Over time, community reference to the packaged evolved to become R-**INLA**. However, installing and loading the package still employs the original name, which may cause some confusion. To install the package, type `install.packages("INLA", repos = "https://inla.r-inla-download.org/R/stable")` into an R console. To load the package, use the R command `library("INLA")`. See https://r-inla.org/ to connect with the community around the development of R-**INLA**, and its application to geostatistics, biostatistics, epidemiology, and econometrics (Lindgren and Rue 2015; Blangiardo and Cameletti 2015).

2. Example data

2.1. Simulated data

The data simulated for Example I (Section 3) and Example II (Section 4) were intended to represent a typical wildlife abundance study. To put the simulated data into context, consider an effort to estimate the abundance of a bird species in a national park, within which are located 72 study sites. At each site, 3 replicate surveys are conducted within 6 weeks, during the peak of the breeding season when birds are most likely to be singing. In order to estimate a trend in abundance over time, clusters of repeated surveys are conducted each breeding season over a 9-year period.

In this scenario, the abundance of the species is thought to vary with two site-level covariates (x_1 and x_2), which represent habitat characteristics at a site and do not change appreciably over time, and a third covariate that indicates the year (x_3). The detection probability is believed to vary according to two covariates (x_1 and x_4). The first covariate for detection, x_1, is the same site-level x_1 that affects abundance, although it has the opposite effect on detection. The other detection covariate, x_4, is a site-survey-year variable that could be related to weather conditions during an individual survey. As is common, due to effects of unknown variables, simulated counts were overdispersed relative to a Poisson distribution. Overdispersed counts were generated and modeled using a negative binomial distribution. Simulation data were generated using the model

$$N_{i,k} \sim \mathrm{NegBin}(\lambda_{i,k}, \theta) \qquad \text{and} \qquad y_{i,j,k} \mid N_{i,k} \sim \mathrm{Bin}(N_{i,k}, p_{i,j,k}), \tag{2}$$

where $\lambda_{i,j}$ was a log-linear function of site and year covariates, as $\log(\lambda_{i,k}) = \beta_0 + \beta_1(x_{1,i}) + \beta_2(x_{2,i}) + \beta_3(x_{3,k})$. Parameter $p_{i,j,k}$ was a logit-linear function of site and site-survey-year covariates, as $\mathrm{logit}(p_{i,j,k}) = \alpha_0 + \alpha_1(x_{1,i}) + \alpha_4(x_{4,i,j,k})$. Parameter values for the linear predictor for λ were set to $\beta_0 = 2$, $\beta_1 = 2$, $\beta_2 = -3$, $\beta_3 = 1$. The overdispersion parameter was set to

```
sim.nmix <- function(n.sites = 72,    # number of study sites
  n.surveys = 3,                      # short-term replicates
  n.years = 9,                        # number of years
  b0 = 2.0,                           # intercept log(lambda)
  b1 = 2.0,                           # x1 slope log(lambda)
  b2 = -3.0,                          # x2 slope log(lambda)
  b3 = 1.0,                           # x3 slope log(lambda)
  a0 = 1.0,                           # intercept logit(p)
  a1 = -2.0,                          # x1 slope logit(p)
  a4 = 1.0,                           # x4 slope logit(p)
  th = 3.0){                          # overdisperison parameter

  # make empty N and Y arrays
  if(n.years %% 2 == 0) {n.years <- n.years + 1}
  N.tr <- array(dim = c(n.sites, n.years))
  Y <- Y.m <- array(dim = c(n.sites, n.surveys, n.years))

  # create abundance covariate values
  x1 <- array(as.numeric(scale(runif(n = n.sites, -0.5, 0.5), scale = FALSE)),
    dim = c(n.sites, n.years))
  x2 <- array(as.numeric(scale(runif(n = n.sites, -0.5, 0.5), scale = FALSE)),
    dim = c(n.sites, n.years))
  yrs <- 1:n.years; yrs <- (yrs - mean(yrs)) / (max(yrs - mean(yrs))) / 2
  x3 <- array(rep(yrs, each = n.sites), dim = c(n.sites, n.years))

  # fill true N array
  lam.tr <- exp(b0 + b1 * x1 + b2 * x2 + b3 * x3)
  for(i in 1:n.sites){
    for(k in 1:n.years){
      N.tr[i, k] <- rnbinom(n = 1, mu = lam.tr[i, k], size = th)
  }}

  # create detection covariate values
  x1.p <- array(x1[,1], dim = c(n.sites, n.surveys, n.years))
  x4 <- array(as.numeric(scale(runif(n = n.sites * n.surveys * n.years,
    -0.5, 0.5), scale = FALSE)), dim = c(n.sites, n.surveys, n.years))

  # average x4 per site-year for example 1
  x4.m <- apply(x4, c(1, 3), mean, na.rm = FALSE)
  out1 <- c()
  for(k in 1:n.years){
    chunk1 <- x4.m[ , k]
    chunk2 <- rep(chunk1, n.surveys)
    out1 <- c(out1, chunk2)
  }
  x4.m.arr <- array(out1, dim = c(n.sites, n.surveys, n.years))

  # fill Y.m count array using x4.m for example 1
  p.tr1 <- plogis(a0 + a1 * x1.p + a4 * x4.m.arr)
  for (i in 1:n.sites){
    for (k in 1:n.years){
      for (j in 1:n.surveys){
        Y.m[i, j, k] <- rbinom(1, size = N.tr[i, k],
          prob = p.tr1[i, j, k])
  }}}
```

Continued on next page.

```
  # fill Y count array using x4 for example 2
  p.tr2 <- plogis(a0 + a1 * x1.p + a4 * x4)
  for (i in 1:n.sites){
    for (k in 1:n.years){
      for (j in 1:n.surveys){
        Y[i, j, k] <- rbinom(1, size = N.tr[i, k], prob = p.tr2[i, j, k])
  }}}

  # format Y.m for data frame output for inla and unmarked
  Y.m.df <- Y.m[ , , 1]
  for(i in 2:n.years){
    y.chunk <- Y.m[ , , i]
    Y.m.df <- rbind(Y.m.df, y.chunk)
  }

  # format covariates for data frame output for inla and unmarked
  x1.df <- rep(x1[ , 1], n.years)
  x2.df <- rep(x2[ , 1], n.years)
  x3.df <- rep(x3[1, ], each = n.sites)
  x1.p.df <- rep(x1.p[ , 1, 1], n.years)
  x4.df <- c(x4.m)

  # put together data frames for inla and unmarked
  inla.df <- unmk.df <- data.frame(y1 = Y.m.df[ , 1], y2 = Y.m.df[ , 2],
    y3 = Y.m.df[ , 3], x1 = x1.df, x2 = x2.df, x3 = x3.df,
    x1.p = x1.p.df, x4.m = x4.df)

  # return all necessary data for examples 1 and 2
  return(list(inla.df = inla.df, unmk.df = unmk.df, n.sites = n.sites,
    n.surveys = n.surveys, n.years = n.years, x1 = x1[ , 1],
    x2 = x2[ , 1], x3 = x3[1, ], x4 = x4, x4.m = x4.m, x4.m.arr =
    x4.m.arr, Y = Y, Y.m = Y.m, lam.tr = lam.tr, N.tr = N.tr, x1.p =
      x1.p[ , 1, 1]
  ))
}
```

Table 1: Source code for the `sim.nmix()` simulation function for Examples I and II.

$\theta = 3$. Parameter values for the linear predictor for p were set to $\alpha_0 = 1$, $\alpha_1 = -2$, $\alpha_4 = 1.0$. All independent variables in the simulation were centered at zero to reduce computational difficulties and to make model intercepts more easily interpreted.

We simulated data for Examples I and II using the `sim.nmix()` function, shown in Table 1, with which we encourage readers to experiment. Parameter and variable names in the function code are similar to those given in the model description, above. Note that the function produces two versions of detection covariate x_4 (`x4` and `x4.m`) and two versions of the count matrix (`Y` and `Y.m`). Covariate `x4` is the same as the site-survey-year variable x_4, described above. It is used to generate `Y`, which is used in Example II. Covariate `x4.m` is derived from `x4`, where values are unique to site and year, but are averaged and duplicated over surveys. It is used to generate `Y.m`, which is employed in Example I. Running `sim.nmix()` results in a list containing data frames for use with R-**INLA** and **unmarked**, and values and vectors for use with **JAGS**. Before running the function, we install and load libraries, and set the seed for the random number generator so that the results are reproducible.

```
R> install.packages("INLA", repos =
+    "https://inla.r-inla-download.org/R/stable")
R> library("INLA")
R> install.packages(c("runjags", "unmarked"))
R> library("runjags")
R> library("unmarked")
R> set.seed(12345)
R> sim.data <- sim.nmix()
```

2.2. Real data

In addition to simulated data, we also demonstrate the use of R-**INLA** and **unmarked** with a real data set in Example III in Section 5. This data set comes from a study by Kéry *et al.* (2005) and is publicly available as part of the **unmarked** package. The data set includes mallard duck (*Anas platyrhynchos*) counts conducted at 239 sites on 2 or 3 occasions during the summer of 2002, as part of a Swiss program that monitors breeding bird abundance (Monitoring Häufige Brutvögel or Swiss Breeding Bird Survey). In addition to counts, the data set also includes 2 site-survey covariates related to detection (survey effort and survey date) and 3 site-level covariates related to abundance (route length, route elevation, and forest cover). Full data set details are given in Kéry *et al.* (2005).

3. Example I

3.1. Goals

In Example I, we demonstrate the use of R-**INLA** and compare use and performance to similar analyses using **JAGS** and **unmarked**. In this first example, the functional forms of R-**INLA**, **JAGS**, and **unmarked** models match the data generating process. Specifically, we used the covariate `x4.m` to generate the count matrix `Y.m`, and analyzed the data with models that use `x4.m` as a covariate. This example was intended to demonstrate the differences and similarities in use, computation time, and estimation results across the three methods when the specified models were the same as the data generating process.

3.2. Analysis with R-INLA

We first analyze the simulated data using the R-**INLA** package. The list returned from the `sim.nmix()` function includes an object called `inla.df`. This object has the following structure.

```
R> str(sim.data$inla.df, digits.d = 2)

'data.frame': 648 obs. of  8 variables:
  $ y1        : int  2 12 25 3 0 3 1 7 2 8 ...
  $ y2        : int  2 22 25 4 1 3 1 11 2 4 ...
  $ y3        : int  4 11 28 2 1 2 0 10 2 3 ...
  $ x1        : num  0.198 0.353 0.238 0.364 -0.066 ...
```

```
 $ x2         : num  -0.159 -0.197 -0.484 0.087 0.429 ...
 $ x3         : num  -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 ...
 $ x1.p       : num  0.198 0.353 0.238 0.364 -0.066 ...
 $ x4.m       : num  0.148 -0.07 0.206 -0.261 -0.046 ...
```

```
R> round(head(sim.data$inla.df), 3)
```

```
    y1   y2   y3      x1      x2    x3    x1.p    x4.m
1    2    2    4   0.198  -0.159  -0.5   0.198   0.148
2   12   22   11   0.353  -0.197  -0.5   0.353  -0.070
3   25   25   28   0.238  -0.484  -0.5   0.238   0.206
4    3    4    2   0.364   0.087  -0.5   0.364  -0.261
5    0    1    1  -0.066   0.429  -0.5  -0.066  -0.046
6    3    3    2  -0.356   0.123  -0.5  -0.356  -0.036
```

This data frame representation of the simulated data has 72 sites × 9 years = 648 rows. Had there only been one year of data, then the data frame would have 72 rows, one per site. The data frame has three columns (y1, y2, and y3) with count data from the count matrix Y.m, one for each of the three replicate surveys within a given year. Had there been six surveys per year, then there would have been six count columns. The three variables thought to affect abundance are represented in columns 4 through 6. Note that, in this scenario, the first two abundance variables are static across years, so there are 72 unique values in a vector that is stacked 9 times. The third abundance variable, the indicator for year, is a sequence of 9 values, where each value is repeated 72 times. It is centered and scaled in this example. The two variables thought to affect detection probability are represented in columns 7 and 8. The first of these variables has the same values as in column 4, so column 7 is a simple copy of column 4. The second of the two detection variables, shown in column 8, varies per site and year in Example I, so there are 648 unique values in this column. Note that any of the covariates for abundance or detection could have varied by site and year, like x4.m.

We made small modifications to this data frame to prepare data for analysis with R-**INLA**. In the code that follows, we use the inla.mdata() function to create an object called counts.and.count.covs. The counts.and.count.covs object is essentially a bundle of information related to the abundance component of the model. Calling the str() function shows that this object is an R-**INLA** list that includes the three count vectors, passed to the function as a matrix, one vector containing the value of 1, which specifies a global intercept for λ, and three vectors corresponding to the covariates for λ. Note that the variable names are standardized by inla.mdata() for computational reasons.

```
R> inla.data <- sim.data$inla.df
R> y.mat <- as.matrix(inla.data[,c("y1", "y2", "y3")])
R> counts.and.count.covs <- inla.mdata(y.mat, 1, inla.data$x1,
+    inla.data$x2, inla.data$x3)
R> str(counts.and.count.covs)
```

```
List of 7
 $ Y1: int [1:648] 2 12 25 3 0 3 1 7 2 8 ...
 $ Y2: int [1:648] 2 22 25 4 1 3 1 11 2 4 ...
```

```
  $ Y3: int [1:648] 4 11 28 2 1 2 0 10 2 3 ...
  $ X1: num [1:648] 1 1 1 1 1 1 1 1 1 1 ...
  $ X2: num [1:648] 0.1983 0.3532 0.2384 0.3636 -0.0661 ...
  $ X3: num [1:648] -0.1595 -0.1966 -0.4842 0.0865 0.429 ...
  $ X4: num [1:648] -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 ...
- attr(*, "class")= chr "inla.mdata"
```

Analysis of N-mixture models with R-**INLA** is accomplished with a call to the `inla()` function. The first argument in the `inla()` call, shown below, is the model formula. On the left side of the formula is the `counts.and.count.covs` object, which includes the vectors of counts, the global intercept for λ, and the covariates related to λ. On the right side of the formula is a 1, to specify a global intercept for p, and the two covariates for p. Note that a wide range of random effects (exchangeable, spatially or temporally structured) for p could be added to the right side of the formula using the `f()` syntax (Rue *et al.* 2017).

The second argument to `inla()` describes the data, provided here as a list that corresponds with the model formula. Third is the likelihood family, which can take values of `"nmix"` for a Poisson-binomial mixture and `"nmixnb"` for a negative binomial-binomial mixture. Run the command `inla.doc("nmix")` for more information on these likelihood families. The fourth (`control.fixed`, for detection parameters) and fifth (`control.family`, for abundance and overdispersion parameters) arguments specify the priors for the two model components. Here, the priors for both abundance and detection parameters are vague normal distributions centered at zero with precision equal to 0.01. The prior for the overdispersion parameter is specified as uniform. Note that a wide variety of other prior distributions are available in R-**INLA**. At the end of the call are arguments to print the progress of model fitting, and to save information that will enable computation of fitted values. Several other characteristics of the analysis can be modified in a call to `inla()`, such as whether or not deviance information criterion (DIC), widely applicable information criterion (WAIC), conditional predictive ordinate (CPO), or probability integral transform (PIT) are computed. See Rue *et al.* (2017) for details.

```
R> out.inla.1 <- inla(counts.and.count.covs ~ 1 + x1.p + x4.m,
+    data = list(counts.and.count.covs = counts.and.count.covs,
+      x1.p = inla.data$x1.p, x4.m = inla.data$x4.m),
+    family = "nmixnb",
+    control.fixed = list(mean = 0, mean.intercept = 0, prec = 0.01,
+      prec.intercept = 0.01),
+    control.family = list(hyper = list(theta1 = list(param = c(0, 0.01)),
+      theta2 = list(param = c(0, 0.01)), theta3 = list(param = c(0, 0.01)),
+      theta4 = list(param = c(0, 0.01)), theta5 = list(prior = "flat",
+      param = numeric()))),
+    verbose = TRUE,
+    control.compute=list(config = TRUE))
R> summary(out.inla.1, digits = 3)

Time used (seconds):
Pre-processing    Running inla    Post-processing    Total
0.421             5.081           0.342              5.844
```

```
Fixed effects:
               mean    sd  0.025quant 0.5quant  0.975quant
(Intercept)  1.053  0.058       0.938     1.054      1.165
x1.p        -1.996  0.197      -2.385    -1.995     -1.611
x4.m         1.056  0.313       0.440     1.056      1.668

The model has no random effects.

Model hyperparameters:
                  mean      sd  0.025quant  0.5quant 0.975quant
beta[1]          2.022   0.034       1.956     2.022      2.090
beta[2]          2.070   0.116       1.839     2.071      2.295
beta[3]         -2.951   0.099      -3.142    -2.953     -2.755
beta[4]          1.142   0.088       0.969     1.142      1.316
overdisp         0.349   0.028       0.296     0.349      0.407
```

Partial output from the `summary()` function, run on the `out.inla.1` object, returned from the `inla()` function, is shown above. The analysis of the model took approximately 6 seconds. Information on the intercept and covariates related to detection are found under the fixed effects section. Note that the posterior median parameter estimates related to the detection intercept (α_0 labeled as `(Intercept)`) and covariates (α_1 as `x1.p` and α_4 as `x4.m`) are very close to, and not significantly different from, input parameter values (Figure 1). Information on abundance and overdispersion parameters are given in the model hyperparameters section. Posterior median estimates for β_0 (labeled as `beta[1]`), β_1 (`beta[2]`), β_2 (`beta[3]`), β_3 (`beta[4]`), and θ (1 / `overdisp` = 2.87) are also very close to, and not significantly different from, input parameter values (Figure 1). Density plots of the full marginal posterior distributions for model parameters (Figure 1) can be viewed using `plot(out.inla.1)`.

The quantity $\lambda_{i,k}$ for each site-year combination in the data set can be computed using covariate values given in the `counts.and.count.covs` object, combined with posterior distributions of parameters in the linear predictor of $\lambda_{i,k}$. Posterior distributions for computed $\lambda_{i,k}$ values can be estimated by repeated sampling from the posteriors of hyperparameters, using the `inla.hyperpar.sample()` function, and repeated solving of the linear predictor. The helper function, `inla.nmix.lambda.fitted()` produces fitted lambda values as described, using the information contained in the model result object. A call to this function, specifying the model result, posterior sample size, and summary output, is as follows.

```
R> out.inla.1.lambda.fits <- inla.nmix.lambda.fitted(result = out.inla.1,
+    sample.size = 5000, return.posteriors = FALSE)$fitted.summary
R> head(out.inla.1.lambda.fits)

  index mean.lambda sd.lambda  q025.lambda  median.lambda   q975.lambda
1     1     10.3329    0.6623       9.0980        10.3109       11.6683
2     2     15.9003    1.1895      13.6760        15.8649       18.3523
3     3     29.2742    2.2882      25.0410        29.2043       34.0370
4     4      7.0490    0.5250       6.0663         7.0270        8.1227
5     5      1.0547    0.0810       0.9072         1.0509        1.2194
6     6      1.4272    0.1031       1.2389         1.4255        1.6353
```

The output from this function call is a summary of estimated posteriors for fitted $\lambda_{i,k}$ values. In this example, there are 648 rows. Comparisons of posterior median fitted $\lambda_{i,k}$ with true simulated $\lambda_{i,k}$ and $N_{i,k}$ values are shown below.

```
R> summary(out.inla.1.lambda.fits$median.lambda)
R> summary(c(sim.data$lam.tr))
```

```
  Min.   1st Qu.    Median     Mean    3rd Qu.       Max.
0.6538    3.4020    7.0070  13.5200    16.0300   123.2000
  Min.   1st Qu.    Median     Mean    3rd Qu.       Max.
0.6834    3.3050    6.8190  13.0300    15.9500   111.3000
```

```
R> cor(out.inla.1.lambda.fits$median.lambda, c(sim.data$lam.tr))
```

```
[1] 0.9986
```

```
R> sum(out.inla.1.lambda.fits$median.lambda)
R> sum(c(sim.data$lam.tr))
R> sum(c(sim.data$N.tr))
```

```
[1] 8758.542
[1] 8444.975
[1] 8960
```

3.3. Analysis with JAGS

Next, we analyzed the same simulated data set using **JAGS**, via the **runjags** package. As for the R-**INLA** analysis, we specified a negative binomial distribution for abundance, vague normal priors for the intercepts and the global effects of the covariates of λ and p, and a flat prior for the overdispersion parameter. The **JAGS** model statement, where the distributions and likelihood function are specified, is shown below for comparison with the arguments to `inla()`.

```
R> jags.model.string <- "
+    model {
+      a0 ~ dnorm(0, 0.01)
+      a1 ~ dnorm(0, 0.01)
+      a4 ~ dnorm(0, 0.01)
+      b0 ~ dnorm(0, 0.01)
+      b1 ~ dnorm(0, 0.01)
+      b2 ~ dnorm(0, 0.01)
+      b3 ~ dnorm(0, 0.01)
+      th ~ dunif(0, 5)
+    for (k in 1:n.years){
+      for (i in 1:n.sites){
+        N[i, k] ~ dnegbin(prob[i, k], th)
```

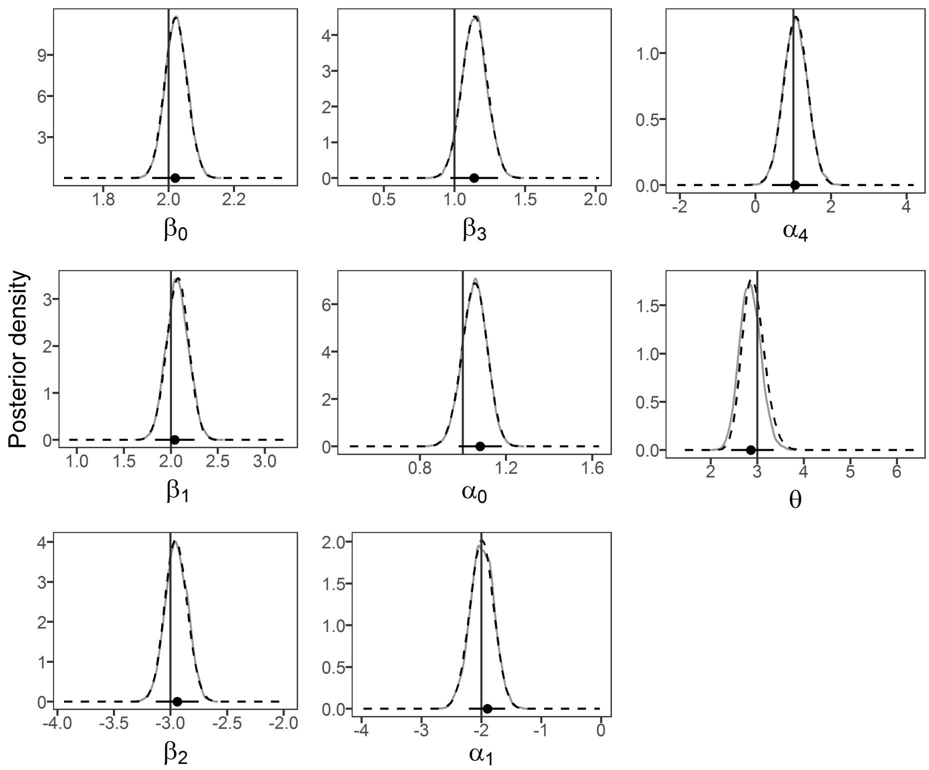

Figure 1: Marginal posteriors of model parameters from R-**INLA** (dashed black lines) and **JAGS** (solid gray lines), along with Maximum Likelihood estimates (black circles) and 95% confidence intervals (horizontal black lines) from **unmarked**. True input values are represented by vertical black lines.

```
+        prob[i, k] <- th / (th + lambda[i, k])
+        log(lambda[i, k]) <- b0 + (b1 * x1[i]) + (b2 * x2[i]) + (b3 * x3[k])
+        for (j in 1:n.surveys){
+          Y.m[i, j, k] ~ dbin(p[i, j, k], N[i, k])
+          p[i, j, k] <- exp(lp[i, j, k]) / (1 + exp(lp[i, j, k]))
+          lp[i, j, k] <- a0 + (a1 * x1.p[i]) + (a4 * x4.m[i, k])
+    }}}}
+  "
```

After specifying the **JAGS** model, we define the parameters to be monitored during the MCMC simulations, bundle numerous values and vectors from the `sim.data` object, and create a function for drawing random initial values for the model parameters. These steps are included in the code supplement, but are not shown here. Finally, we set the run parameters, such as the number of chains and iterations, and start the MCMC process. Run parameters were chosen such that MCMC diagnostics indicated converged chains (potential scale reduction factors < 1.05) and reasonably robust posterior distributions (effective sample sizes $> 3,000$). Note that the recommended number of effective samples for particularly robust inference is closer to 6,000 (Gong and Flegal 2016). Thus, MCMC processing times reported here could be considered optimistic estimates. The MCMC simulation is initiated with a call to run.jags(). Partial output from the simulation, related to parameter estimates, is shown below.

```
R> out.jags.1 <- run.jags(model = jags.model.string, data = jags.data,
+    monitor = params, n.chains = 3, inits = inits, burnin = 3000,
+    adapt = 3000, sample = 6000, thin = 10, modules = "glm on",
+    method = "parallel")
R> round(summary(out.jags.1), 3)[ , c(1:5, 9, 11)]
```

```
            Lower95 Median Upper95   Mean    SD SSeff psrf
a0            0.941  1.053   1.161  1.053 0.057  3155    1
a1           -2.380 -1.990  -1.618 -1.994 0.195  2954    1
a4            0.434  1.053   1.661  1.052 0.314  5695    1
b0            1.956  2.024   2.089  2.024 0.034  7065    1
b1            1.851  2.070   2.301  2.071 0.115  6028    1
b2           -3.142 -2.946  -2.755 -2.947 0.099 17539    1
b3            0.969  1.142   1.315  1.142 0.089 18000    1
th            2.401  2.840   3.295  2.850 0.230 18000    1
```

Similar to the R-**INLA** analysis, median parameter estimates from the **JAGS** model were close to, and not significantly different from, the input values used to generate the data (Figure 1). The potential scale reduction factor for all variables was < 1.05, and the effective sample size for all variables was approximately $3,000$ or greater. The simulation ran in parallel on 3 virtual cores, 1 MCMC chain per core, and took approximately $2,960$ seconds.

3.4. Analysis with unmarked

Lastly, we prepared the simulated data for the **unmarked** analysis, which involved slight modification of the unmk.df object created using the sim.nmix() function. As with the **JAGS** analysis, these steps are included in the code supplement, but are not illustrated here. The **unmarked** analysis was run by a call to the pcount() function. The first argument in the call to pcount() is the model formula, which specifies the covariates for detection and then the covariates for abundance. This is followed by an argument identifying the unmarked data object and the form of the mixture model, negative binomial-binomial in this case.

```
R> out.unmk.1 <- pcount(~ 1 + x1.p + x4.m ~ 1 + x1 + x2 + x3,
+    data = unmk.data, mixture = "NB")
R> summary(out.unmk.1)
```

```
Abundance (log-scale):
            Estimate     SE     z   P(>|z|)
(Intercept)     2.02 0.0321  62.9  0.00e+00
x1              2.04 0.1071  19.0  1.69e-80
x2             -2.94 0.0982 -29.9  5.36e-20
x3              1.14 0.0882  13.0  2.26e-38

Detection (logit-scale):
```

```
            Estimate       SE       z     P(>|z|)
(Intercept)     1.08   0.0507   21.27   2.29e-100
x1.p           -1.90   0.1576  -12.08    1.30e-33
x4.m            1.04   0.3102    3.35    8.02e-04

Dispersion (log-scale):
Estimate         SE     z    P(>|z|)
1.05         0.0804  13.1   4.91e-39
```

Maximum likelihood estimates for model parameters from **unmarked** were also close to, and not significantly different from, input values (Figure 1). Note that the dispersion estimate, after exponentiation, was 2.86. The **unmarked** estimates were produced in approximately 86 seconds.

3.5. Example I summary

Example I demonstrated basic use of R-**INLA** to analyze N-mixture models and highlighted similarities and differences between it and two other commonly used approaches. In demonstrating the use of R-**INLA**, we showed that the input data format is not too complicated, and that the formatting process can be accomplished with a few lines of code. Similarly, model specification uses a straightforward extension of the standard syntax in R, where the counts and covariates for λ are specified through an R-**INLA** object included on the left side of the formula, and fixed covariates and random effects for p are specified on the right side of the formula. The data format and model specification syntax of R-**INLA** is not too different from **unmarked**, whereas those of both packages are considerably different from **JAGS** and other MCMC software, such as **OpenBUGS**, **WinBUGS**, and Stan.

Regarding performance, R-**INLA**, **JAGS**, and **unmarked** all successfully extracted simulation input values. Figure 1 shows marginal posterior distributions produced by R-**INLA** and **JAGS**, and estimates and 95% confidence intervals from **unmarked**. These results derive from data from one random manifestation of the input values. Thus, we do not expect the posterior distributions for the estimates to be centered at the input values, which would be expected if the simulation was repeated many times. However, we do expect the input values to fall somewhere within the posterior distributions and 95% confidence limits, which is what occurred here. Figure 1 shows that, for similarly specified models, R-**INLA** (dashed black lines) and **JAGS** (solid gray lines) yielded practically identical marginal posterior distributions for model parameters. Figure 1 also illustrates the general agreement between the credible intervals associated with R-**INLA** and **JAGS** and the confidence intervals associated with **unmarked**.

Where R-**INLA**, **JAGS**, and **unmarked** differed substantially was in computing time. In this example, R-**INLA** took 6 seconds, **JAGS** took $2,960$ seconds, and **unmarked** took 86 seconds to produce results. Thus, R-**INLA** was approximately 500 times faster than **JAGS** and 10 times faster than **unmarked**. This was the case despite the fact that **unmarked** produced ML estimates and the **JAGS** analysis was run in parallel with each of three MCMC chains simulated on a separate virtual computing core. If parallel computing had not been used with **JAGS**, processing the **JAGS** model would have taken approximately twice as long. If MCMC simulations were run until effective sample sizes of $6,000$ were reached, processing time would have doubled again.

In sum, when compared to other tools, R-**INLA** is relatively easy to implement and produces accurate estimates of Bayesian posteriors very quickly. Its utility depends on the degree to which the data generating process can be captured accurately in model specification. However, as mentioned above, certain N-mixture models cannot be specified using R-**INLA**. For the data in Example I, the count matrix was produced using a detection covariate that was averaged to the site-year level. This averaged covariate was subsequently specified in the model. But what happens when the site-survey-year covariate is an important component of the data generating process, and it can't be entered into the analysis model in this form? This is the question explored in Example II.

4. Example II

4.1. Goals

In Example II, we show the consequences of not being able to specify a site-survey-year covariate for detection, under a range of conditions. We conducted a Monte Carlo experiment where, for each iteration, the count matrix for the analysis, `Y`, was generated with the `sim.nmix()` function using the site-survey-year covariate `x4`. The count data were then analyzed with two **JAGS** models. The first model incorporated the site-survey-year `x4` covariate. The second model incorporated the averaged site-year `x4.m`, instead. For each iteration, we randomly varied the size of α_4 when generating the simulated data. We expected that the simpler model, with `x4.m`, would yield biased estimates when the magnitude of α_4 was relatively large, and unbiased estimates when the magnitude of α_4 was relatively small. All computing code related to Example II is given in the supplemental code file.

4.2. Analysis with JAGS

Parameter values entered into `sim.nmix()`, other than those for α_4, were the same as those used in Example I. Similarly, the **JAGS** model specification, other than parts associated with α_4, was the same as that used in Example I. Given the long processing time associated with **JAGS** models in Example I, we only ran and saved 1000 MCMC simulations (no thinning, after 500 adaptive and 100 burn-in iterations) during each of the 50 Monte Carlo runs in Example II. This number is not sufficient for drawing inference from marginal posteriors, but was sufficient for looking at qualitative patterns in posterior medians. For each of these runs, a value for α_4 was drawn from a uniform distribution that ranged from -3 to 3. Parameter bias was represented for each model parameter as the difference between the simulation input and the posterior median estimated value. The results of the simulations are depicted in Figure 2.

4.3. Example II summary

Even with as few as 50 Monte Carlo runs, it was apparent that biases in parameter estimates increased with the magnitude of α_4 (Figure 2). When the magnitude of α_4 was small, with an absolute value less than 1, the bias was negligible. When the magnitude of α_4 was large, with an absolute value greater than 2, the bias was considerable (Figure 2). When interpreting the effect size, bear in mind that `x4` ranged from -0.5 to 0.5.

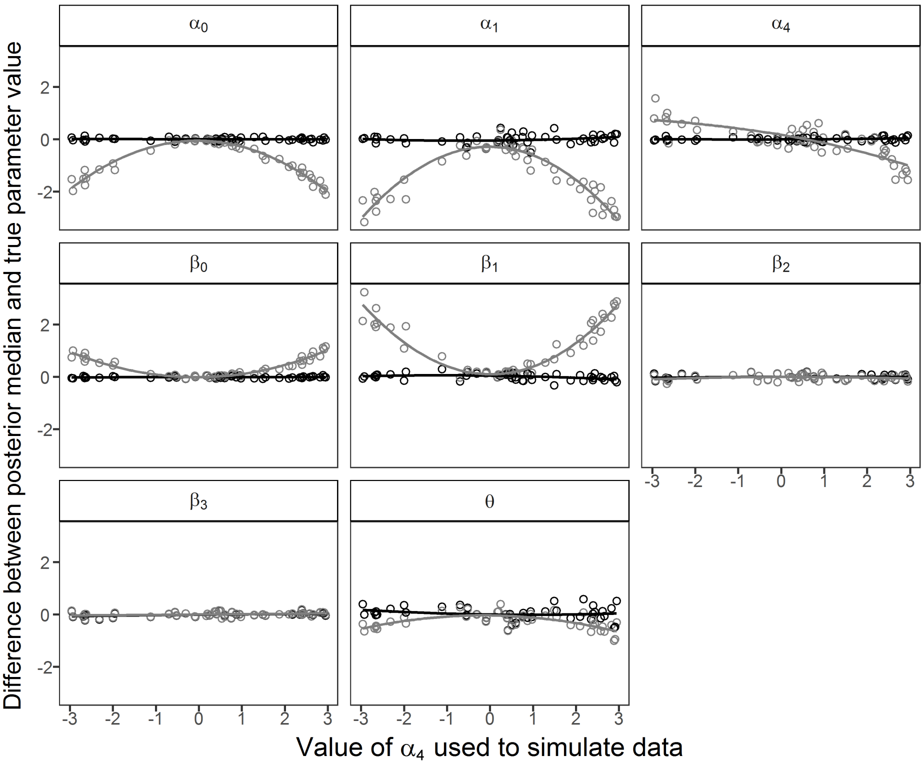

Figure 2: Differences between posterior median parameter values and true input parameter values as a function of the α_4 value used to simulate data. Black circles and lines are from the model with the site-survey-year covariate, `x4`, and gray circles and lines are from the model with an averaged site-year covariate, `x4.m`. Parameter name is given in the strip across the top of each panel.

5. Example III

5.1. Goals

In Example III, we explore the performance of R-**INLA** using real data, a publicly available data set of mallard duck counts from Switzerland during 2002. By employing real data, we hoped to evaluate (i) the performance of R-**INLA** using data that were not predictable by design and (ii) the practical consequences of not being able to specify site-survey covariates in R-**INLA**. The data set is available as a demonstration data set in **unmarked**, so we compared the performance of R-**INLA** with that of **unmarked** using the analysis settings and model structure described in **unmarked** documentation.

5.2. Analysis with R-INLA

The mallard data are provided in the **unmarked** package as a list with three components: a matrix of counts (`mallard.y`), a list of matrices of detection covariates (`mallard.obs`), and a data frame of abundance covariates (`mallard.site`). Data in **unmarked** are organized in structures called unmarked frames, which are viewed as a data frame when printed.

```
R> data("mallard")
R> mallard.umf <- unmarkedFramePCount(y = mallard.y, siteCovs =
+    mallard.site, obsCovs = mallard.obs)
R> mallard.umf[1:6, ]

Data frame representation of unmarkedFrame object.
   y1   y2   y3    elev length forest  ivel1  ivel2  ivel3
1   0    0    0  -1.173  0.801 -1.156 -0.506 -0.506 -0.506
2   0    0    0  -1.127  0.115 -0.501 -0.934 -0.991 -1.162
3   3    2    1  -0.198 -0.479 -0.101 -1.136 -1.339 -1.610
4   0    0    0  -0.105  0.315  0.008 -0.819 -0.927 -1.197
5   3    0    3  -1.034 -1.102 -1.193  0.638  0.880  1.042
6   0    0    0  -0.848  0.741  0.917 -1.329 -1.042 -0.899
...
    date1  date2  date3
1  -1.761  0.310  1.381
2  -2.904 -1.047  0.596
3  -1.690 -0.476  1.453
4  -2.190 -0.690  1.239
5  -1.833  0.167  1.381
6  -2.619  0.167  1.381
```

As discussed above, it is not possible to take advantage of survey-level covariates when analyzing N-mixture models with R-**INLA**. So, before analysis with R-**INLA**, we averaged the survey-level variables, `ivel` and `date`, per site using the `rowMeans()` function.

```
R> length <- mallard.site[ , "length"]
R> elev <- mallard.site[ , "elev"]
R> forest <- mallard.site[ , "forest"]
R> mean.ivel <- rowMeans(mallard.obs$ivel, na.rm = TRUE)
R> mean.ivel[is.na(mallard.ivel)] <- mean(mallard.ivel, na.rm = TRUE)
R> mean.date <- rowMeans(mallard.obs$date, na.rm = TRUE)
R> mean.date.sq <- mean.date^2
R> mallard.inla.df <- data.frame(y1 = mallard.y[ , "y.1"],
+    y2 = mallard.y[ , "y.2"], y3 = mallard.y[ , "y.3"],
+    length, elev, forest, mean.ivel, mean.date, mean.date.sq)
R> round(head(mallard.inla.df), 3)

  y1 y2 y3 length   elev forest mean.ivel mean.date mean.date.sq
1  0  0  0  0.801 -1.173 -1.156    -0.506    -0.023        0.001
2  0  0  0  0.115 -1.127 -0.501    -1.029    -1.118        1.251
3  3  2  1 -0.479 -0.198 -0.101    -1.362    -0.238        0.056
4  0  0  0  0.315 -0.105  0.008    -0.981    -0.547        0.299
5  3  0  3 -1.102 -1.034 -1.193     0.853    -0.095        0.009
6  0  0  0  0.741 -0.848  0.917    -1.090    -0.357        0.127
```

The data are now in a format that can be analyzed readily using R-**INLA**. The data frame representation has 239 sites × 1 year = 239 rows, one column for each replicate count, and one column for each detection and abundance covariate. Once in this form, it is easy to create an `inla.mdata()` object and run the analysis. In preparing `counts.and.count.covs`, we specify an intercept and effects of transect length (`length`), elevation (`elev`), and forest cover (`forest`) on abundance. In the `model` argument to `inla()`, we specify an intercept and effects of survey intensity (`ivel`) and survey date (`date`) on detection. As before, the data argument is a list that corresponds with the model formula. The family argument specifies a negative binomial-binomial mixture. The prior for each intercept and covariate was specified as vague normal distributions, and that for the overdispersion parameter was specified as a uniform distribution.

```
R> counts.and.count.covs <- inla.mdata(mallard.y, 1, length, elev, forest)
R> out.inla.2 <- inla(counts.and.count.covs ~ 1 + mean.ivel +
+    mean.date + mean.date.sq,
+    data = list(counts.and.count.covs = counts.and.count.covs,
+      mean.ivel = mean.ivel, mean.date = mean.date, mean.date.sq =
+      mean.date.sq),
+    family = "nmixnb",
+    control.family = list(hyper = list(theta5 = list(prior = "flat",
+      param = numeric()))))
R> summary(out.inla.2, digits = 3)
```

A portion of the summary for `out.inla.2` is shown below. Note that posterior summaries described in the fixed effects section pertain to the intercept and covariates of logit(p). In the hyperparameters section, `beta[1]`, `beta[2]`, `beta[3]`, and `beta[4]` identify posterior summaries for the log(λ) intercept, and transect length, elevation, and forest cover effects.

```
Fixed effects:
               mean    sd 0.025quant  0.5quant  0.975quant
(Intercept)  -0.397 0.383     -1.170    -0.389       0.335
mean.ivel     0.039 0.212     -0.378     0.039       0.455
mean.date    -1.044 0.433     -1.923    -1.036      -0.195
mean.date.sq -0.318 0.304     -0.962    -0.301       0.233

The model has no random effects.

Model hyperparameters:
             mean     sd  0.025quant  0.5quant  0.975quant
beta[1]    -1.412  0.296      -1.966    -1.424      -0.801
beta[2]    -0.290  0.190      -0.664    -0.291       0.086
beta[3]    -0.998  0.318      -1.595    -1.011      -0.341
beta[4]    -0.771  0.203      -1.178    -0.767      -0.382
overdisp    1.228  0.264       0.799     1.194       1.837
```

5.3. Analysis with unmarked

The **unmarked** frame, `mallard.umf`, created above, can be used directly by the `pcount()` function in **unmarked**. The data and model structure described in the `pcount()` function below is similar to that used above in the R-**INLA** analysis, except for one key difference: here, `ivel` and `date` are site-survey level variables instead of the site-level means used in the R-**INLA** analysis.

```
R> out.unmk.2 <- pcount(~ ivel + date + I(date^2) ~ length + elev + forest,
+    mixture = "NB", data = mallard.umf)
R> summary(out.unmk.2)

Abundance (log-scale):
             Estimate     SE        z    P(>|z|)
(Intercept)    -1.786  0.281   -6.350   2.15e-10
length         -0.186  0.214   -0.868   3.86e-01
elev           -1.372  0.293   -4.690   2.73e-06
forest         -0.685  0.216   -3.166   1.54e-03

Detection (logit-scale):
             Estimate     SE        z    P(>|z|)
(Intercept)    -0.028  0.285   -0.099      0.921
ivel            0.174  0.227    0.766      0.444
date           -0.313  0.147   -2.132      0.033
I(date^2)      -0.005  0.081   -0.059      0.953

Dispersion (log-scale):
Estimate         SE      z   P(>|z|)
-0.695        0.364  -1.91     0.056
```

5.4. Example III summary

Comparing the results, we see that the 95% credible intervals for parameter estimates from the R-**INLA** analysis overlapped broadly with the 95% confidence intervals from the **unmarked** analysis, so parameter estimates were not significantly different from one another (Figure 3). Regardless of technique, the same set of parameters had estimates significantly different from zero (Figure 3), and significant effects were of the same magnitude and direction in both analyses. Using both techniques, detection decreased as the season progressed, and abundance decreased with increasing forest cover and elevation. Parameter estimates and biological conclusions were similar despite the fact that site-survey detection covariates were used for **unmarked** and site-averaged detection covariates were used for R-**INLA**. Note that **unmarked** estimated moderate effects of detection covariates which, according to the results in Example II, would indicate that parameter estimates from the R-**INLA** analysis were not substantially biased. These conclusions may have been different given a different data set, where detection covariates had very strong effects or were not otherwise controlled by survey design.

6. Discussion

The purpose of this work was to detail the use of the R-**INLA** package (Rue *et al.* 2017) to analyze N-mixture models and to compare analyses using R-**INLA** to two other common approaches: **JAGS** (Plummer 2003; Lunn, Jackson, Best, Thomas, and Spiegelhalter 2012), via

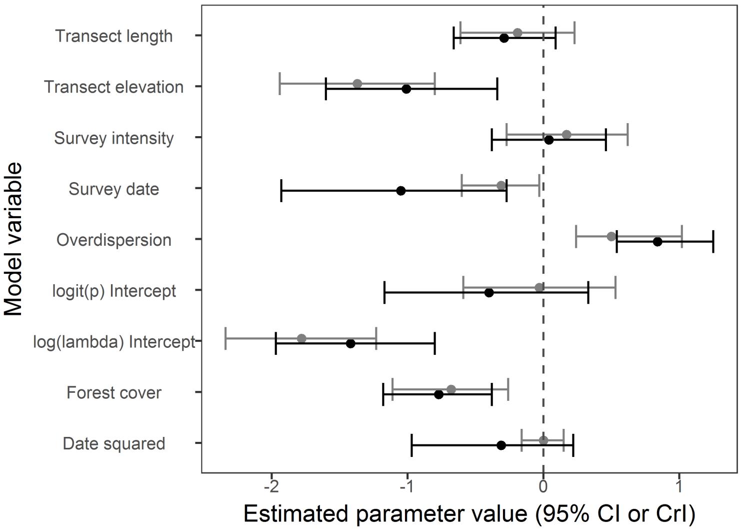

Figure 3: Parameter estimates and 95% confidence intervals from **unmarked** (gray circles and lines) and posterior medians and 95% credible intervals from R-**INLA** (black circles and lines) from an N-mixture model analysis of mallard duck abundance. Model parameters are identified by their associated variable names listed on the vertical axis. The **unmarked** model included site-survey covariates for survey intensity and survey date, while the R-**INLA** model included site-averaged versions. A value of zero (no effect) is depicted by the vertical dashed gray line.

the **runjags** package (Denwood 2016), which employs MCMC methods and allows Bayesian inference, and the **unmarked** package (Fiske and Chandler 2011), which uses maximum likelihood and allows frequentist inference. While we selected **JAGS** as the representative MCMC approach, we expect that our conclusions would be qualitatively similar for other MCMC software, such as **OpenBUGS**, **WinBUGS**, or Stan. We are not aware of other commonly-used software for analyzing N-mixture models in a maximum likelihood framework, besides **unmarked**.

Comparisons showed that R-**INLA** can be a complementary tool in the wildlife biologist's analytical tool kit. Strengths of R-**INLA** include Bayesian inference, based on highly accurate approximations of posterior distributions, which were derived roughly 500 times faster than MCMC methods, where models are specified using a syntax that should be familiar to R users, and where data are formatted in a straightforward way with relatively few lines of code. The straightforward model syntax and data format could help lower barriers to adoption of N-mixture models for biologists who are not committed to learning BUGS or Stan syntax. The substantial decrease in computation time should facilitate use of a wider variety of model and variable selection techniques (e.g., cross validation and model averaging), ones that are not commonly used in an MCMC context due to practical issues related to computing time (Kéry and Schaub 2011).

Limitations of R-**INLA** are mainly related to the more restricted set of N-mixture models that can be specified. Of the approaches described here, ones that use MCMC allow users ultimate flexibility in specifying models. For example, with **JAGS**, site-survey covariates for detection are possible, multiple types of mixed distributions are available (Joseph *et al.* 2009; Martin *et al.* 2011), and a variety of random effects can be specified for both λ and p (Kéry and Schaub 2011). In comparison, the current version of R-**INLA** does not handle site-survey covariates, employs only Poisson-binomial and negative binomial-binomial mixtures, and handles random effects for p only. A practical consequence of the random effects limitation is that, while site and site-year posteriors for λ can be estimated using R-**INLA**, site and site-year posteriors for N are not currently available (but see Appendix). In cases where site-survey covariates are particularly important, and not otherwise controlled by survey design, where different mixed distributions are required, or where random effects associated with λ are needed, an MCMC approach appears to be most appropriate (Figure 2).

When compared to **unmarked**, the R-**INLA** approach is similar in regards to familiar model syntax and data format. The approaches are also similar in that both yield results much faster than MCMC, enabling a richer set of options in terms of model and variable selection. The two approaches differ in that R-**INLA** is approximately 10 times faster than **unmarked**, likely due to the different method used to compute model likelihoods (see Appendix). They also differ in that **unmarked** can accommodate site-survey covariates, whereas R-**INLA** does not, and that R-**INLA** can accommodate random effects for p, whereas **unmarked** does not. In cases where both computing speed and specification of site-survey covariates are critical, **unnmarked** appears to be an appropriate tool.

In conclusion, R-**INLA**, **JAGS** (and **WinBUGS**, **OpenBUGS**, and Stan), and **unmarked** all allow users to analyze N-mixture models for estimating wildlife abundance while accounting for imperfect detection. Each method has its strengths and limitations. R-**INLA** appears to be an attractive option when survey-level covariates are not essential, familiar model syntax and data format are desired, Bayesian inference is preferred, and fast computing time is required.

Acknowledgments

We thank C. Burkhalter, T. Onkelinx, U. Halekoh, E. Pebesma, and an anonymous reviewer for commenting on previous drafts of this manuscript.

References

Barker RJ, Schofield MR, Link WA, Sauer JR (2018). "On the Reliability of N-Mixture Models for Count Data." *Biometrics*, **74**(3), 369–377. doi:10.1111/biom.12734.

Blangiardo M, Cameletti M (2015). *Spatial and Spatio-Temporal Bayesian Models with R-INLA*. John Wiley & Sons, West Sussex. doi:10.1002/9781118950203.

Carpenter B, Gelman A, Hoffman M, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1), 1–32. doi:10.18637/jss.v076.i01.

Chandler RB, Royle JA, King DI (2011). "Inference About Density and Temporary Emigration in Unmarked Populations." *Ecology*, **92**(7), 1429–1435. doi:10.1890/10-2433.1.

Dail D, Madsen L (2011). "Models for Estimating Abundance from Repeated Counts of an Open Metapopulation." *Biometrics*, **67**(2), 577–587. doi:10.1111/j.1541-0420.2010.01465.x.

Dénes FV, Silveira LF, Beissinger SR (2015). "Estimating Abundance of Unmarked Animal Populations: Accounting for Imperfect Detection and Other Sources of Zero Inflation." *Methods in Ecology and Evolution*, **6**(5), 543–556. doi:10.1111/2041-210x.12333.

Denwood MJ (2016). "**runjags**: An R Package Providing Interface Utilities, Model Templates, Parallel Computing Methods and Additional Distributions for MCMC Models in **JAGS**." *Journal of Statistical Software*, **71**(9). doi:10.18637/jss.v071.i09.

Dodd CK, Dorazio RM (2004). "Using Counts to Simultaneously Estimate Abundance and Detection Probabilities in a Salamander Community." *Herpetologica*, **60**(4), 468–478. doi:10.1655/03-60.

Fiske I, Chandler R (2011). "**unmarked**: An R Package for Fitting Hierarchical Models of Wildlife Occurrence and Abundance." *Journal of Statistical Software*, **43**(10), 1–23. doi:10.18637/jss.v043.i10.

Gong L, Flegal JM (2016). "A Practical Sequential Stopping Rule for High-Dimensional Markov Chain Monte Carlo." *Journal of Computational and Graphical Statistics*, **25**(3), 684–700. doi:10.1080/10618600.2015.1044092.

Joseph LN, Elkin C, Martin TG, Possingham HP (2009). "Modeling Abundance Using N-Mixture Models: The Importance of Considering Ecological Mechanisms." *Ecological Applications*, **19**(3), 631–642. doi:10.1890/07-2107.1.

Kéry M (2010). *Introduction to **WinBUGS** for Ecologists: Bayesian Approach to Regression, ANOVA, Mixed Models and Related Analyses.* Academic Press, Burlington.

Kéry M, Dorazio RM, Soldaat L, Van Strien A, Zuiderwijk A, Royle JA (2009). "Trend Estimation in Populations with Imperfect Detection." *Journal of Applied Ecology*, **46**(6), 1163–1172. doi:10.1111/j.1365-2664.2009.01724.x.

Kéry M, Royle JA (2010). "Hierarchical Modelling and Estimation of Abundance and Population Trends in Metapopulation Designs." *Journal of Animal Ecology*, **79**(2), 453–461. doi:10.1111/j.1365-2656.2009.01632.x.

Kéry M, Royle JA (2015). *Applied Hierarchical Modeling in Ecology: Analysis of Distribution, Abundance and Species Richness in R and BUGS: Volume 1: Prelude and Static Models.* Academic Press, Burlington.

Kéry M, Royle JA, Schmid H (2005). "Modeling Avian Abundance from Replicated Counts Using Binomial Mixture Models." *Ecological Applications*, **15**(4), 1450–1461. doi:10.1890/04-1120.

Kéry M, Schaub M (2011). *Bayesian Population Analysis Using **WinBUGS**: A Hierarchical Perspective.* Academic Press, Burlington.

Korner-Nievergelt F, Roth T, von Felton S, Guélat J, Almasi B, Korner-Nievergelt P (2015). *Bayesian Data Analysis in Ecology Using Linear Models with R, BUGS, and Stan.* Academic Press, London.

Lindgren F, Rue H (2015). "Bayesian Spatial Modeling with R-**INLA**." *Journal of Statistical Software*, **63**(19), 1–25. doi:10.18637/jss.v063.i19.

Lindgren F, Rue H, Lindstrom J (2011). "An Explicit Link between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach." *Journal of the Royal Statistical Society B*, **73**(4), 423–498. doi:10.1111/j.1467-9868.2011.00777.x.

Lunn D, Jackson C, Best N, Thomas A, Spiegelhalter D (2012). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Chapman & Hall/CRC, Boca Raton.

Martin J, Royle JA, Mackenzie DI, Edwards HH, Kéry M, Gardner B (2011). "Accounting for Non-Independent Detection When Estimating Abundance of Organisms with a Bayesian Approach." *Methods in Ecology and Evolution*, **2**(6), 595–601. doi:10.1111/j.2041-210x.2011.00113.x.

O'Donnell KM, Thompson FR, Semlitsch RD (2015). "Partitioning Detectability Components in Populations Subject to within-Season Temporary Emigration Using Binomial Mixture Models." *PLoS ONE*, **10**(3), e0117216. doi:10.1371/journal.pone.0117216.

Plummer M (2003). "**JAGS**: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In K Hornik, F Leisch, A Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Technische Universität Wien, Vienna, Austria. URL https://www.R-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf.

Pollock KH, Nichols JD, Simons TR, Farnsworth GL, Bailey LL, Sauer JR (2002). "Large Scale Wildlife Monitoring Studies: Statistical Methods for Design and Analysis." *Environmetrics*, **13**(2), 105–119. doi:10.1002/env.514.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Royle JA (2004). "*N*-Mixture Models for Estimating Population Size from Spatially Replicated Counts." *Biometrics*, **60**(1), 108–115. doi:10.1111/j.0006-341x.2004.00142.x.

Royle JA, Dorazio RM (2008). *Hierarchical Modeling and Inference in Ecology: The Analysis of Data from Populations, Metapopulations, and Communities*. Academic Press, Burlington.

Royle JA, Nichols JD (2003). "Estimating Abundance from Repeated Presence-Absence Data or Point Counts." *Ecology*, **84**(3), 777–790. doi:10.1890/0012-9658(2003)084[0777:eafrpa]2.0.co;2.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392. doi:10.1111/j.1467-9868.2008.00700.x.

Rue H, Riebler A, Sørbye SH, Illian JB, Simpson DP, Lindgren FK (2017). "Bayesian Computing with R-**INLA**: A Review." *Annual Reviews of Statistics and Its Applications*, **4**(3), 395–421. doi:10.1146/annurev-statistics-060116-054045.

Spiegelhalter DJ, Thomas A, Best NG, Lunn D (2003). ***WinBUGS** Version 1.4 User Manual*. MRC Biostatistics Unit, Cambridge. URL http://www.mrc-bsu.cam.ac.uk/bugs/.

Thomas A (2014). ***OpenBUGS** Version 3.2.3.* **OpenBUGS** Foundation. URL http://www.openbugs.net/.

Wenger SJ, Freeman MC (2008). "Estimating Species Occurrence, Abundance, and Detection Probability Using Zero-Inflated Distributions." *Ecology*, **89**(10), 2953–2959. doi:10.1890/07-1127.1.

Wu G, Holan SH, Nilon CH, Wikle CK (2015). "Bayesian Binomial Mixture Models for Estimating Abundance in Ecological Monitoring Studies." *Annals of Applied Statistics*, **9**(1), 1–26. doi:10.1214/14-aoas801.

Yoccoz NG, Nichols JD, Boulinier T (2001). "Monitoring of Biological Diversity in Space and Time." *Trends in Ecology and Evolution*, **16**(8), 446–453. doi:10.1016/s0169-5347(01)02205-4.

A. Posterior probability for N

Currently, it is not possible to extract posteriors for N when analyzing N-mixture models using R-**INLA**. This functionality, which would utilize output from the `inla.posterior.sample()` function, could be available in future versions based on the following logic. Assume the Poisson for N, such that $\mathsf{P}(N \mid \lambda) = p_0(N; \lambda)$, and

$$\mathsf{P}(y_1, \ldots, y_m \mid N) = \sum \left[\prod_{i=1}^{m} \mathrm{Bin}(y_i; N, p) \right] p_0(N; \lambda), \tag{3}$$

where $y_1, \ldots, y_m = \Upsilon$, and $N \geq \max(y_1, \ldots, y_m)$. If we have samples from the posterior of $(p, \lambda) \mid \Upsilon$, we can compute the posterior marginal of $N \mid \Upsilon$ as follows. If (p, λ) is fixed, then

$$\mathsf{P}(N \mid \Upsilon) \propto \left[\prod_{i=1}^{m} \mathrm{Bin}(y_i; N, p) \right] p_0(N; \lambda), \tag{4}$$

and this expression is evaluated for $N = \max(y_1, \ldots, y_m)$, and renormalized. We can integrate out $(p, \lambda) \mid \Upsilon$ using samples from the posteriors, as

$$\mathsf{P}(N \mid \Upsilon) = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{Z(p_j, \lambda_j)} \left[\prod_{i=1}^{M} \mathrm{Bin}(y_i; N, p_j) \right] p_0(N; \lambda_j), \tag{5}$$

for M samples $(p_1, \lambda_1), \ldots, (p_M, \lambda_M)$ from the posterior of $(p, \lambda) \mid \Upsilon$. That is, we average the probability for each N, renormalize, and normalize for each sample by computing $Z(p, \lambda)$.

B. Recursive computations of the 'nmix' likelihood

The likelihood for the simplest case is

$$\mathrm{Prob}(y) = \sum_{N=y}^{\infty} \mathrm{Pois}(N; \lambda) \; \times \; \mathrm{Bin}(y; N, p) \tag{6}$$

where $\mathrm{Pois}(N; \lambda)$ is the density for the Poisson distribution with mean λ, $\lambda^N \exp(-\lambda)/N!$, and $\mathrm{Bin}(y; N, p)$ is the density for the binomial distribution with N trials and probability p, $\binom{N}{y} p^y (1-p)^{N-p}$. Although the likelihood can be computed directly when replacing the infinite limit with a finite value, we will demonstrate here that we can easily evaluate it using a recursive algorithm that is both faster and more numerically stable. The same idea is also applicable to the negative binomial case, and the case where we have replicated observations of the same N. We leave it to the reader to derive these straight forward extensions.

The key observation is that both the Poisson and the binomial distribution can be evaluated recursively in N,

$$\mathrm{Pois}(N; \lambda) = \mathrm{Pois}(N-1; \lambda) \frac{\lambda}{N} \tag{7}$$

and

$$\mathrm{Bin}(y; N, p) = \mathrm{Bin}(y; N-1, p) \frac{N}{N-y} (1-p), \tag{8}$$

and then also for the Poisson-binomial product

$$\mathrm{Pois}(N;\lambda)\ \mathrm{Bin}(y;N,p) = \mathrm{Pois}(N-1;\lambda)\ \mathrm{Bin}(y;N-1,p)\frac{\lambda}{N-y}(1-p). \tag{9}$$

If we define $f_i = \lambda(1-p)/i$ for $i = 1, 2, \ldots$, we can make use of this recursive form to express the likelihood with a finite upper limit as

$$\begin{aligned} \mathrm{Prob}(y) &= \sum_{N=y}^{N_{\max}} \mathrm{Pois}(N;\lambda)\ \mathrm{Bin}(y;N,p) \\ &= \mathrm{Pois}(y;\lambda)\ \mathrm{Bin}(y;y,p)\Big\{1 + f_1 + f_1 f_2 + \ldots + f_1 \ldots f_{N_{\max}}\Big\} \\ &= \mathrm{Pois}(y;\lambda)\ \mathrm{Bin}(y;y,p)\Big\{1 + f_1(1 + f_2(1 + f_3(1 + \ldots)))\Big\} \end{aligned}$$

The log-likelihood can then be evaluated using the following simple R code.

```
R> fac <- 1; ff <- lambda * (1-p)
R> for(i in (N.max - y):1) fac <- 1 + fac * ff / i
R> log.L <- dpois(y, lambda, log = TRUE) +
+    dbinom(y, y, p, log = TRUE) + log(fac)
```

Since this evaluation is recursive in decreasing N, we have to choose the upper limit $N_{\max}$ in advance, for example as an integer larger than y so that $\frac{\lambda(1-p)}{N_{\max}-y}$ is small. Note that we are computing `fac` starting with the smallest contributions, which are more numerically stable.

Affiliation:

Timothy D. Meehan
National Audubon Society
Boulder, Colorado, United States of America
E-mail: tmeehan@audubon.org

Nicole L. Michel
National Audubon Society
San Francisco, California, United States of America
E-mail: nmichel@audubon.org

Håvard Rue
King Abdulla University of Science and Technology
Thuwal, Saudi Arabia
E-mail: haavard.rue@kaust.edu.sa

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2020, Volume 95, Issue 2 *Submitted:* 2017-04-26
doi:10.18637/jss.v095.i02 *Accepted:* 2018-10-29