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

Timothy D. Meehan[*], *Nicole L. Michel*[†], *and Håvard Rue*[‡]

April 7, 2017

1. Introduction

1.1 Background

Successful management of wildlife species requires accurate estimates of abundance (Yoccoz et al. 2001). One common method for estimating animal abundance is direct counts (Pollock et al. 2002). Efforts to obtain accurate abundance estimates via direct counts can be hindered by both the cryptic nature of many wildlife species and by other factors such as observer expertise, weather, and habitat structure. The lack of perfect detection in wildlife surveys is common, and can cause abundance to be greatly underestimated (Wenger and Freeman 2008, Joseph et al. 2009).

In recent years, new sampling schemes and modeling approaches have enabled improved estimates of animal abundance that are less biased by non-detection (Denes et al. 2015). One such sampling scheme, termed the "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, or reproduction), then the ratio of detections to non-detections during repeated counts can inform an estimate of detection probability. This detection probability can, in effect, be used to correct abundance estimates for imperfect detection.

Data resulting from this sampling scheme are often modeled using an explicitly hierarchical statistical model referred to as an N-mixture model (Royle and Nichols 2003, Dodd and Dorazio 2004, Royle 2004, Kéry et al. 2005). A simple form of an N-mixture model, a binomial mixture model, describes 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, lambda (λ). Specifically,

$$N_i \sim Poisson(\lambda) \qquad and \qquad Y_{i,j}|N_i \sim binomial(N_i, p).$$

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 $logit(p_{i,j}) = \alpha_0 + \alpha_1 * x_{i,j}$, a logit-linear function of site-survey covariates.

These estimation approaches can be extended to cover K distinct breeding seasons, which correspond with distinct years for annually-breeding wildlife species (Kéry et al. 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 $N_{i,k} \sim Poisson(\lambda_{i,k})$, $\quad Y_{i,j,k}|N_{i,k} \sim binomial(N_{i,k}, p_{i,k})$. Like the single-year specification, λ is commonly modeled using site and site-year covariates, and p

[*]National Audubon Society, Boulder, Colorado USA
[†]National Audubon Society, San Francisco, California USA
[‡]King Abdulla University of Science and Technology, Thuwal SAU

using site-survey-year covariates. There are other variations of N-mixture models that accommodate overdispersed counts through use of a negative binomial distribution (Kéry and Royle 2010), zero-inflated Poisson distribution (Wenger and Freeman 2008), or observation-level random intercepts (Kéry and Schaub 2011). Yet other variations account for non-independent detection probabilities through use of a beta-binomial distribution (Martin et al. 2011), parse different components of detection through the use of unique covariates (O'Donnell et al. 2015), or relax assumptions of population closure (Chandler et al. 2011, Dail and Madsen 2011). We do not discuss all of these variations here, but refer interested readers to Denes et al. (2015) for a nice overview.

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. Much of the development of N-mixture models, and many of the examples in the wildlife literature, implement models using Bayesian modeling software such as WinBUGS, OpenBUGS, or JAGS (Lunn et al. 2012). These are extremely powerful and flexible platforms for analyzing Bayesian models, but they come with a few important challenges. First, many wildlife biologists are not accustomed to coding statistical models using the BUGS modeling syntax. While there are several outstanding resources aimed at teaching this skill (Royle and Dorazio 2008, Kéry 2010, Kéry and Schaub 2011, Kéry and Royle 2015) it 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, and may not occur at all.

There are other tools available for analyzing N-mixture models that alleviate these practical issues. The unmarked package (Fiske et al. 2011) for R statistical computing software (R Core Team 2016) offers several options for analyzing N-mixture models within a frequentist framework, with the capacity to accommodate overdispersion and dynamic populations. The model coding syntax used in unmarked is a simple extension of the standard R syntax. Models are analyzed using a Maximum Likelihood (ML) approach, so model analysis is often completed in a fraction of the time taken using an MCMC approach. 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 – the loss of the intuitive inferential framework associated with Bayesian analysis.

Here we discuss analysis of N-mixture models using the R-INLA package (Rue et al. 2013) 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 et al. 2009). INLA was developed to allow estimation of posteriors in a fraction of the time taken by MCMC. Like unmarked, the model syntax used in 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. The R-INLA approach is different from unmarked in that inference about model parameters falls within a Bayesian paradigm.

1.2 Overall objectives

The purpose of this manuscript is to demonstrate analysis of N-mixture models using the R-INLA package. In the process, we employ simulated and real count datasets, and analyze them using JAGS, via the runjags package (Denwood 2016) for R, the unmarked package (Fiske et al. 2011), and the R-INLA package (Rue et al. 2013). In each case, we demonstrate how data are formatted, how models are specified, how model estimates compare to simulation inputs, and how methods compare in terms of computational performance. We also explore a limitation of the R-INLA approach, related to model specification. Specifically, while it is possible to specify survey level covariates for detection using JAGS and unmarked, this is not possible using 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-level differences in survey conditions should they occur. However, in the process of averaging, important information related to detection is discarded, which could lead to biased detection and abundance estimates under certain conditions.

Much of the code used to conduct analyses is shown in the body of this manuscript. However, some repeated code, and code related to generating tables and figures, is not shown for brevity. All code can be accessed via https://github.com/tmeeha/inlaNMix.

1.3 Simulated data

The data simulated for this analysis were intended to represent a typical wildlife study. To put the simulation in 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 likely to be singing. In order to estimate a trend in abundance over time, these repeated surveys are conducted over a 9-year period.

In this scenario, the abundance of the species is thought to vary with two site-level covariates (λ covariates 1 and 2) that represents habitat characteristics at a site and do not change appreciably over time. The detection probability is also believed to vary according to two covariates (p covariates 1 and 3). The first covariate for detection, p covariate 1, is the same site-level covariate 1 that affects abundance, although it has the opposite effect on detection. The other detection covariate, p covariate 3, is a site-survey-year variable that could be related to weather conditions during a given count.

As is commonly the case, we assume that the counts are overdispersed due to the effects of unknown variables. Overdispersion is generated and modeled using a negative binomial distribution for the count component of the model. The specific model parameters and their simulated values are: p.b0 = 1.0 (p intercept), p.b1 = -2.0 (effect of p covariate 1), p.b3 = 1.0 (effect of p covariate 3), lam.b0 = 2.0 (λ intercept), lam.b1 = 2.0 (effect of λ covariate 1), lam.b2 = -3.0 (effect of λ covariate 2), lam.b4 = 1.0 (effect of year on λ), disp.size = 3.0 (size of the overdispersion parameter). All independent variables in the simulation vary are centered at zero to alleviate computational difficulties and to make model intercepts more easily interpreted.

We simulated data for this study using the following function. Note that the function produces two versions of detection covariate 3, x.p.3 and x.p.3.arr, and two versions of the count matrix, y1 and y2. x.p.3 is the site-survey-year variable described above. It is used to generate y2, which is used in Example II. x.p.3.mean is derived from x.p.3, where values are unique to site and year, but averaged over surveys. It is used to generate y1, which is used in Example I.

```r
data4sim <- function(n.sites = 72,    # number study sites
                     n.surveys = 3,   # short term replicates
                     n.years = 9,     # number years, MAKE ODD NUMBER
                     lam.b0 = 2.0,    # intercept for log lambda
                     lam.b1 = 2.0,    # slope for log lambda, covariate 1
                     lam.b2 = -3.0,   # slope for log lambda, covariate 2
```

```
                    lam.b4 = 1.0,   # slope for log lambda, year
                    p.b0 = 1.0,     # intercept for logit p
                    p.b1 = -2.0,    # slope for logit p, covariate 1
                    p.b3 = 1.0,     # slope for logit p covariate 3
                    disp.size = 3.0 # size of the overdisperison
                    ){
# setup
if(n.years %% 2 == 0) {n.years <- n.years + 1}     # make years odd
N.tr <- array(dim = c(n.sites, n.years))           # array for true abund
y1 <- array(dim = c(n.sites, n.surveys, n.years))  # array for eg1 counts
y2 <- array(dim = c(n.sites, n.surveys, n.years))  # array for eg2 counts
# abundance covariate values
x.lam.1 <- array(as.numeric(scale(runif(n=n.sites, -0.5, 0.5), scale=F)),
                 dim=c(n.sites, n.years)) # site-level covariate 1
x.lam.2 <- array(as.numeric(scale(runif(n=n.sites, -0.5, 0.5), scale=F)),
                 dim=c(n.sites, n.years)) # site-level covariate 1
yrs <- 1:n.years; yrs <- (yrs - mean(yrs)) / (max(yrs - mean(yrs))) / 2
yr <- array(rep(yrs, each=n.sites), dim=c(n.sites, n.years)) # std years
# fill abundance array
lam.tr <- exp(lam.b0 + lam.b1*x.lam.1 + lam.b2*x.lam.2 + lam.b4*yr)   # true lam
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 = disp.size) # true N
}}
# detection covariate values
x.p.1 <- array(x.lam.1[,1], dim=c(n.sites,n.surveys,n.years))
x.p.3 <- array(as.numeric(scale(runif(n=n.sites*n.surveys*n.years, -0.5, 0.5),
                               scale=F)), dim=c(n.sites,n.surveys,n.years))
# average x.p.3 per site-year
x.p.3.mean <- apply(x.p.3, c(1,3), mean, na.rm=F)
out1 <- c()
for(k in 1:n.years){
 chunk1 <- x.p.3.mean[,k]
 chunk2 <- rep(chunk1, n.surveys)
 out1 <- c(out1, chunk2)
}
x.p.3.arr <- array(out1, dim=c(n.sites, n.surveys, n.years))
# fill count array with site-yr x.p.3
p.tr1 <- plogis(p.b0 + p.b1*x.p.1 + p.b3*x.p.3.arr) # true p
for (i in 1:n.sites){
  for (k in 1:n.years){
    for (j in 1:n.surveys){
      y1[i,j,k] <- rbinom(1, size=N.tr[i,k], prob=p.tr1[i,j,k])
}}}
# fill count array with site-surv-yr x.p.3
p.tr2 <- plogis(p.b0 + p.b1*x.p.1 + p.b3*x.p.3)      # true p
for (i in 1:n.sites){
```

```r
    for (k in 1:n.years){
      for (j in 1:n.surveys){
        y2[i,j,k] <- rbinom(1, size=N.tr[i,k], prob=p.tr2[i,j,k])
  }}}
  # return data
  return(list(n.sites=n.sites, n.surveys=n.surveys, n.years=n.years,
              x.p.1=x.p.1[,1,1], x.p.3=x.p.3, x.p.3.mean=x.p.3.mean,
              x.p.3.arr=x.p.3.arr, x.lam.1=x.lam.1[,1],
              x.lam.2=x.lam.2[,1], yr=yr[1,], y1=y1, y2=y2,
              lam.tr=lam.tr, N.tr=N.tr))
} #end
```

```r
# create dataset for exercise 1 and 2
sim.data <- data4sim()
```

1.4 Real data

In addition to simulated data, we also demonstrate the use of R-INLA using a real dataset in Example III. This dataset comes from a study by Kéry et al. (2005) and is publicly available as part of the unmarked package. The dataset 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). In addition to counts, the dataset also includes 2 site-survey covariates related to detection (survey effort and survey date), and 3 site level covariates related to abundance (route length, elevation, and forest cover). Full dataset details are given in Kéry et al. (2005).

2. Example I

2.1 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 exercise, the forms of JAGS, unmarked, and R-INLA models match the data generating process. Specifically, we used the covariate x.p.3.mean to generate the count matrix y1 and analyzed the data using models that use x.p.3.mean as a covariate. This exercise is intended to demonstrate the differences and similarities in use, computation time, and estimation results across the three packages when the specified model is reasonably close to the data generating process.

2.2 Analysis with JAGS

We first analyze the simulated data using JAGS. In defining the model, we specify a negative binomial model for the abundance component, and use vague normal priors for the intercepts and the global effects of the covariates of λ and p.

```
jags.model.string <- "
model {

  # priors
  intP ~ dnorm(0, 0.01)       # detection intercept
  bCov1P ~ dnorm(0, 0.01)     # detection cov 1 effect
  bCov3P ~ dnorm(0, 0.01)     # detection cov 3 effect
  intLam ~ dnorm(0, 0.01)     # lambda intercept
  bCov1Lam ~ dnorm(0, 0.01)   # lambda cov 1 effect
  bCov2Lam ~ dnorm(0, 0.01)   # lambda cov 2 effect
  bYr ~ dnorm(0, 0.01)        # year effect
  overDisEst ~ dunif(0, 5)    # overdispersion size

  # abundance component
  for (k in 1:nYears){
    for (i in 1:nSites){
      N[i, k] ~ dnegbin(prob[i, k], overDisEst) # negative binomial specification
      prob[i, k] <- overDisEst / (overDisEst + lambda[i, k]) # overdispersion effect
      log(lambda[i, k]) <- intLam + (bCov1Lam * x.lam.1[i]) + (bCov2Lam * x.lam.2[i]) +
                          (bYr * yr[k])

  # detection component
      for (j in 1:nSurveys){
        y[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] <- intP + (bCov1P * x.p.1[i]) + (bCov3P * x.p.3[i, k])
      } # close j loop
    } # close i loop
  } # close k loop

} # close model loop
"
```

Next, we define the parameters to be monitored during the MCMC runs, bundle the data for JAGS, and create a function for drawing random initial values for the model parameters. The initial values for abundance are made to avoid values of “NA” and zero, as these would cause computational problems.

```
# parameters to monitor
params <- c("intP", "bCov1P", "bCov3P", "intLam", "bCov1Lam","bCov2Lam",
            "bYr", "overDisEst")
# jags data
jags.data <- list(y = sim.data$y1, x.lam.1 = sim.data$x.lam.1,
             x.lam.2 = sim.data$x.lam.2, yr = sim.data$yr,
             x.p.1 = sim.data$x.p.1, x.p.3 = sim.data$x.p.3.mean,
             nSites = sim.data$n.sites, nSurveys = sim.data$n.surveys,
             nYears = sim.data$n.years)
# initial values
```

```r
N.init <- sim.data$y1 # initial count values
N.init[is.na(N.init)] <- 1 # clean up NA's
N.init <- apply(N.init, c(1, 3), max) + 1 # zero values cause trouble
inits <- function() list(N = N.init, intLam = rnorm(1, 0, 0.01),
                     intP = rnorm(1, 0, 0.01), bCov1P = rnorm(1, 0, 0.01),
                     bCov2Lam = rnorm(1,0,0.01), bCov1Lam = rnorm(1, 0, 0.01),
                     bCov3P = rnorm(1, 0, 0.01), bYr = rnorm(1, 0, 0.01),
                     overDisEst = runif(1, 0.5, 2.5))
```

Finally, we set the run parameters and start the MCMC process. Run parameters were chosen such that MCMC diagnostics indicated converged chains (Gelman-Rubin statistics < 1.05) and reasonably robust posterior distributions (effective sample sizes > 1000). Note that the recommended number of effective samples for particularly robust inference is closer to 6000 (Gong and Flegal 2016). So MCMC processing times reported here could be considered low estimates.

```r
# set run parameters
nc <- 3; na <- 2500; nb <- 2500; ni <- 5000; nt <- 10
# run jags
ptm <- proc.time()
out.jags <- run.jags(model = jags.model.string, data = jags.data,
                     monitor = params, n.chains = nc, inits = inits,
                     burnin = nb, adapt = na, sample = ni, thin = nt,
                     modules = "glm on", method = "parallel")
```

Mean parameter estimates from the JAGS model are reasonably close to, and not significantly different from, the input values used to generate the data (Fig. 1). The potential scale reduction factor for all variables was < 1.01, and the effective sample size for all variables was > 2117. The simulation ran in parallel on 3 virtual cores, 1 MCMC chain per core, and took approximately 2162 seconds.

2.3 Analysis with unmarked

Next, we prepare the simulated data for the unmarked analysis, which involves slight modification of the data in the sim.data object. Here, the count data was changed from a 3-dimensional $I * J * K$ array to a 2-dimensional $I * K$ row by J column matrix. Each static, site level variable was duplicated and stacked K times to form a single-column vector. A column vector was created to identify each year in the stacked data. The site-year variable, x.p.3.mean, was transformed from 2-dimensional matrix to a single column vector. Reformatted variables were then assembled in an unmarked data structure called an unmarked frame.

```r
# format count data
y.unmk <- sim.data$y1[ , , 1]
for(i in 2:sim.data$n.years){
  y.chunk <- sim.data$y1[ , , i]
  y.unmk <- rbind(y.unmk, y.chunk)
}
# format covariates
x.lam.1.unmk <- rep(sim.data$x.lam.1, sim.data$n.years)
```

```r
x.lam.2.unmk <- rep(sim.data$x.lam.2, sim.data$n.years)
yr.unmk <- rep(sim.data$yr, each = sim.data$n.sites)
x.p.1.unmk <- rep(sim.data$x.p.1, sim.data$n.years)
x.p.3.unmk <- c(sim.data$x.p.3.mean)
site.covs.unmk <- data.frame(x.lam.1.unmk, x.lam.2.unmk, yr.unmk,
                             x.p.1.unmk, x.p.3.unmk)
# make unmarked pcount frame
unmk.data <- unmarkedFramePCount(y = y.unmk, siteCovs = site.covs.unmk)
```

The first argument in the call to the pcount() function was the model formula, which specified the covariates for detection and then the covariates for abundance. This was followed by an argument identifying the appropriate unmarked frame, and the form of the count portion of the mixture model, negative binomial.

```r
# run unmk model
ptm <- proc.time()
out.unmk <- pcount(~ x.p.1.unmk + x.p.3.unmk
                   ~ x.lam.1.unmk + x.lam.2.unmk + yr.unmk,
                    data = unmk.data, mixture = "NB")
```

Maximum likelihood estimates from the unmarked analysis are also close to, and not significantly different from, input values (Fig. 1) The unmarked estimates were produced in approximately 72 seconds.

2.4 Analysis with R-INLA

Next, we prepare data for R-INLA, which uses data formats similar to those used by unmarked. The object counts.and.count.covs is an R-INLA object that includes an $I * K$ row by J column matrix of counts. Next comes a value of 1, which is turned into an $I * K$ length vector of ones to specify that the model has a global intercept for λ. Finally, there are two $I * K$ length vector of values for the two static, site covariates of abundance, where the vector of values for I sites is stacked K times. In addition to the counts.and.count.covs object, we also define x.p.1.inla, which is a copy of x.lam.1.inla, and x.p.3.inla, which is the $I * K$ length vector corresponding with x.p.3.mean.

```r
# format count data
y.inla <- sim.data$y1[ , , 1]
for(i in 2:sim.data$n.years){
  y.chunk <- sim.data$y1[ , , i]
  y.inla <- rbind(y.inla, y.chunk)
}
# format covariates
x.lam.1.inla <- rep(sim.data$x.lam.1, sim.data$n.years)
x.lam.2.inla <- rep(sim.data$x.lam.2, sim.data$n.years)
yr.inla <- rep(sim.data$yr, each = sim.data$n.sites)
x.p.1.inla <- rep(sim.data$x.p.1, sim.data$n.years)
x.p.3.inla <- c(sim.data$x.p.3.mean)
# make inla.mdata object
counts.and.count.covs <- inla.mdata(y.inla, 1, x.lam.1.inla, x.lam.2.inla, yr.inla)
```

The call to the inla() function includes several components. First is the model statement. On the left side of the formula is the counts.and.count.covs object that includes the matrix of counts 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 fixed covariates for p. Note that a wide range of random effects for p could be added to the right side of the formula (Rue et al. 2013). The second argument 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. The fourth and fifth arguments specify the priors for the two model components. In this case, the priors are specified similarly to those in the JAGS model, but a variety of other options are available. Several other characteristics of the analysis can be modified using the call to inla(). See Rue et al. (2013) for details.

```
# run inla model
ptm <- proc.time()
out.inla <- inla(counts.and.count.covs ~ 1 + x.p.1.inla + x.p.3.inla,
         data = list(counts.and.count.covs = counts.and.count.covs,
                     x.p.1.inla = x.p.1.inla, x.p.3.inla = x.p.3.inla),
         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)))))
```

Mean parameter estimates from the R-INLA model are also close to, and not significantly different from, input values (Fig. 1) Full marginal posterior distributions from R-INLA are also shown in Fig. 1. The analysis took approximately 7 seconds.

2.5 Example I summary

Example I demonstrated basic use of R-INLA and highlighted similarities and differences between it and two other commonly used approaches. In demonstrating the use of R-INLA, we show that the input data format is not complicated, and that the formatting process can be accomplished with relatively few lines of code. Similarly, model specification uses a straightforward extension of the standard syntax in R, where the count matrix and covariates for λ are specified through an R-INLA object included on the left side of the formula, and covariates for p are specified on the right side of the formula. Data format and model specification syntax of R-INLA are not too different from unmarked, which are both considerably different from JAGS and other BUGS-oriented MCMC software.

Regarding performance, R-INLA, unmarked, and JAGS successfully extracted simulation input values. Fig. 1 depicts marginal posterior distributions produced by JAGS and R-INLA, and 95% confidence intervals from unmarked. These results derive from data resulting 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 intervals, which is what occurs here. Figure 1 shows that, for similarly specified

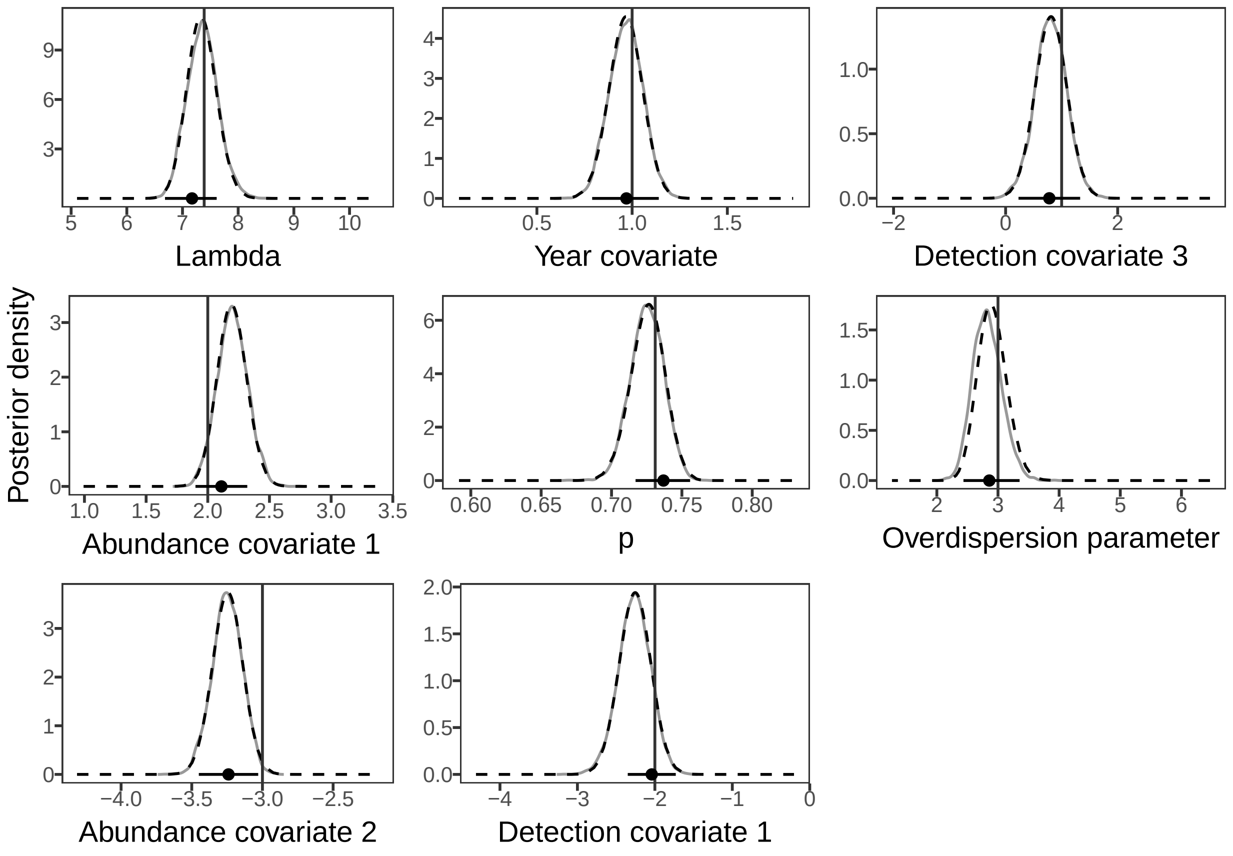

Figure 1: Marginal posteriors from JAGS (solid gray lines) and R-INLA (dashed black lines), along with maximum liklihood estimates (black circles) and 95% confidence intervals (horizontal black lines) from unmarked. True input values are represented by vertical black lines.

models, R-INLA (dashed black lines) and JAGS (solid gray lines) yielded practically identical marginal posterior distributions for model parameters.

Where R-INLA, unmarked, and JAGS differed was in computing time. In this example, R-INLA took approximately 7 seconds, unmarked took approximately 72 seconds, and JAGS took 2,162 seconds to produce results. Thus, R-INLA was approximately 10 times faster than unmarked and 200 times faster than JAGS. This was the case despite the fact that unmarked uses ML algorithms and the JAGS model was run in parallel with each of three MCMC chains simulated on a separate virtual 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 an effective sample size of 6000 was reached, processing times would have been tripled again.

In sum, when compared to other tools, R-INLA is relatively easy to implement and produces accurate estimates of Bayesian posteriors relatively quickly. This statement assumes that the data generating process can be captured accurately in model specification. However, as mentioned above, certain models can not 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 used in the analysis. But what happens when the site-survey-year covariate is an important component of the data generating process, and it can't be specified in the model in this form? This is the question explored in Example II.

3. Example II

3.1 Goals

In Example 2, we show the consequences of not being able to specify a site-survey-year covariate under a range of conditions. Specifically, we conducted a Monte Carlo simulation where, for each iteration, the count matrix for the analysis, y2, was generated with the data generating function described in 1.3, using the site-survey-year covariate x.p.3. The count data were then analyzed with two JAGS models. One model incorporated an averaged site-survey x.p.3.mean as a covariate, such as in 2.1. The other model incorporated the site-survey-year covariate as a fixed effect instead. In each iteration we varied the size of the effect for x.p.3. We suspected that the simpler model, with x.p.3.mean, would yield biased estimates when the effect of x.p.3 was large and unbiased estimates when the effect was small.

3.2 Analysis with JAGS

Given the long processing time associated with JAGS models, we only ran 3000 iterations (after 1000 adaptive iterations and 1000 burn-in iterations) during each simulation. This number is not sufficient for making inference on marginal posteriors, but was sufficient for looking at qualitative patterns in posterior means. In each of the 50 iterations the size of the effect for x.p.3 was drawn from a uniform distribution that ranged from -3 to 3. The results of the simulations are depicted in Fig. 2.

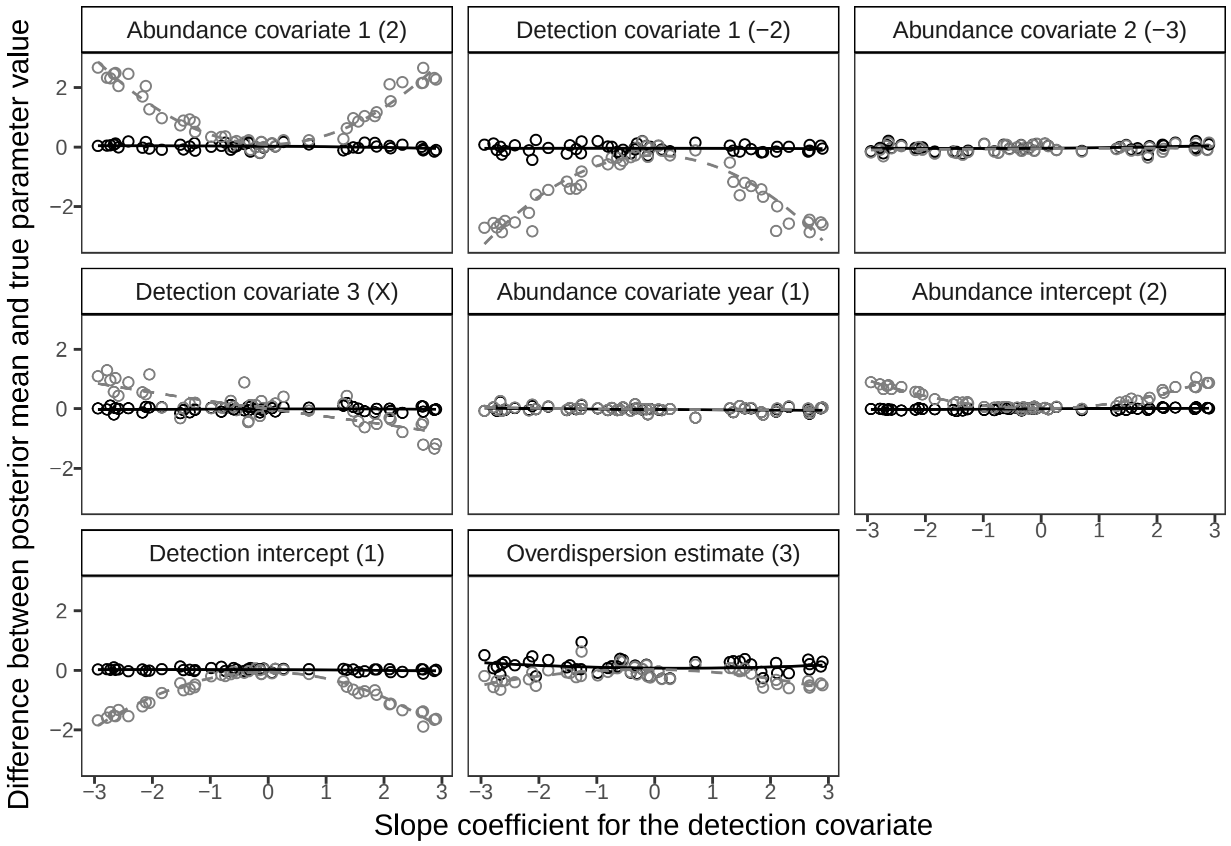

Figure 2: Differences between the estimated posterior mean values and the true input parameter values as a function of the slope coefficient (effect size) for the site-survey-year covariate, x.p.3. Black circles and lines are output from the model with the site-survey-year covariate, x.p.3, and gray circles and lines are from the model with an averaged site-year covariate, x.p.3.mean. Parameter name and true input value is given in the strip across the top of each panel.

3.3 Example II summary

As expected, biases in estimates of abundance (λ intercept) and the effects of abundance covariates (specifically λ covariate 2) increased with the effect strength of the site-survey-year covariate for p. When the effect size was small, with an absolute value less than 1, the bias was minimal. When the effect size was large, with an absolute value greater than 2, the bias was large. When interpreting the effect size, bear in mind that x.p.3 ranged from -0.5 to 0.5.

4. Example III

4.1 Goals

In Example III, we explored the performance of R-INLA using real data, a publicly available dataset on mallard duck abundance in Switzerland in 2002. By incorporating real data, we hoped to evaluate the performance of R-INLA using data that were not predictable by design and the practical impacts of not being able to specify survey level covariates in R-INLA. The dataset is available as a demonstration dataset in unmarked, so we compared the performance of R-INLA and unmarked in Example III.

4.1 Analysis with unmarked

The unmarked model was specified with linear effects of transect length, elevation, and forest cover on abundance, and a linear effect and quadratic effect of sampling intensity and sampling date, respectively, on detection. Both sampling intensity and sampling date were site-survey covariates.

```r
# get real mallard data from unmarked package
data(mallard)
# format unmarked data
mallard.y.unmk <- mallard.y
mallard.site.unmk <- mallard.site
mallard.obs.unmk <- mallard.obs
mallardUMF <- unmarkedFramePCount(mallard.y.unmk, siteCovs = mallard.site.unmk,
                                  obsCovs = mallard.obs.unmk)
# run unmarked model
mallard.out.unmk <- pcount(~ ivel+ date + I(date^2) ~ length + elev + forest,
                           mixture = "NB", mallardUMF, K=30)
```

4.2 Analysis with INLA

The R-INLA model used a similar model structure. The only difference was that the R-INLA model was not able to process site-survey covariates for detection, so averaged site level covariates were used, instead.

```r
# format inla data
mallard.y.inla <- mallard.y
mallard.length.inla <- mallard.site[,2]
mallard.elev.inla <- mallard.site[,1]
mallard.forest.inla <- mallard.site[,3]
mallard.ivel.inla <- rowMeans(mallard.obs$ivel, na.rm=T) # average per site
mallard.ivel.inla[is.na(mallard.ivel.inla)] <- mean(mallard.ivel.inla, na.rm=T)
mallard.date.inla <- rowMeans(mallard.obs$date, na.rm=T) # average per site
mallard.date2.inla <- mallard.date.inla^2
# make inla.mdata object
counts.and.count.covs <- inla.mdata(mallard.y.inla, 1, mallard.length.inla,
                                    mallard.elev.inla, mallard.forest.inla)
# run inla model
mallard.out.inla <- inla(counts.and.count.covs ~ 1 + mallard.ivel.inla +
                           mallard.date.inla + mallard.date2.inla,
         data = list(counts.and.count.covs = counts.and.count.covs,
                     mallard.ivel.inla = mallard.ivel.inla,
                     mallard.date.inla = mallard.date.inla,
                     mallard.date2.inla = mallard.date2.inla),
         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)))))
```

The 95% credible intervals from the R-INLA model overlapped considerably with the 95% confidence intervals from unmarked model, so parameter estimates were not significantly different from one another. Also, the parameters with values significantly different from zero were the same regardless of technique. Using both techniques, detection decreased linearly as the season progressed and abundance decreased as linear functions of increased forest cover and elevation gain. 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 both approaches estimated moderate effects of detection covariates which, according to the results in Example II, might lead to similar results regardless of which covariate is used. Conclusions may have been different with a different real dataset where detection covariates had very strong effects.

5. Discussion

The purpose of this study was to demonstrate the use of the R-INLA package to analyze N-mixture models and to compare performance of R-INLA to two other common approaches, JAGS (via the runjags package), which uses MCMC methods and allows Bayesian inference (Lunn et al. 2013), and unmarked, which uses maximum likelihood and allows frequentist inference (Fiske et al. 2011).

Results demonstrate 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 are derived 200 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

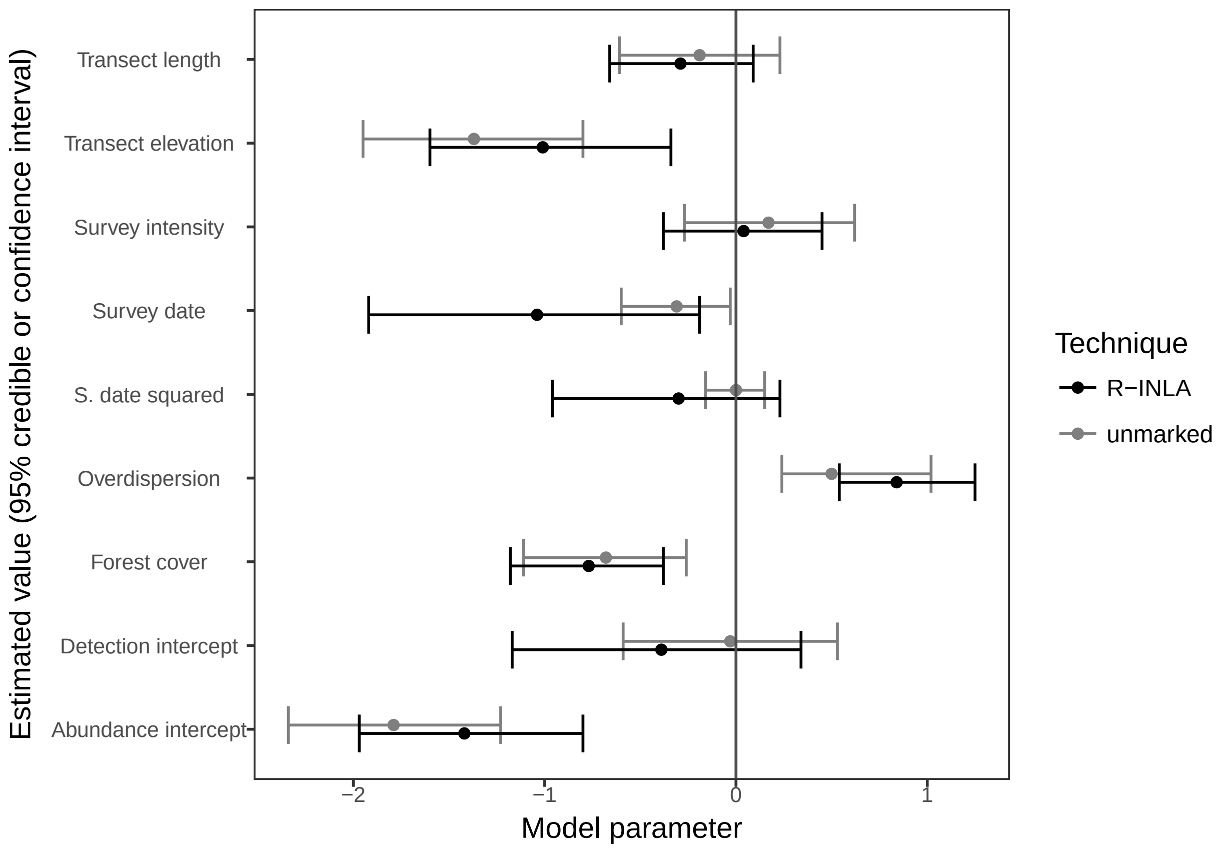

Figure 3: Estimates and 95% confidence intervals from unmarked (gray circles and lines) and posterior means and 95% credible intervals from R-INLA (black lines) from an N-mixture model of mallard duck abundance. The unmarked model specified unique site-survey covariates for survey effort and survey date, while the R-INLA model specified site-average versions. A value of zero is depicted by the vertical gray line.

data format could help remove barriers to adoption of N-mixture models for biologists who are not committed to learning the BUGS syntax. The substantial decrease in computation time should allow a wider variety of model and variable selection techniques (e.g., cross validation and model averageing), 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 explored here, BUGS-based approaches allow users ultimate flexibility in specifying models. For example, with JAGS, site-survey-year 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 survey-level covariates, employs only Poisson-binomial and negative binomial-binomial mixtures, and handles random effects (exchangeable, spatially and temporally structured) for p only. In cases where survey-level covariates are particularly strong and not controlled in the sampling design, R-INLA will not be the appropriate tool (Fig. 2).

When compared to unmarked, the R-INLA approach is similar in regards to model syntax and data format. The approaches are also similar in that both yield results much faster than MCMC. The two approaches differ in that R-INLA is roughly 10 times faster than unmarked, possibly due to the different approach used to compute model likelihoods (see Appendix). They also differ in that unmarked can accomodate survey level covariates and can analyze dynamic N-mixture models (Dail and Madsen 2011).

In conclusion, JAGS (and WinBUGS, OpenBUGS), unmarked, and R-INLA all allow users to employ 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, fast computing time is needed, and Bayesian inference is preferred.

6. Appendix

Recursive computations of the likelihood for N-mixture models

The likelihood for the simplest case is

$$\text{Prob}(y) = \sum_{n=y}^{\infty} \text{Poisson}(n; \lambda) \ \times \ \text{binomial}(y; n, p),$$

where Poisson$(n; \lambda)$ is the density for the Poisson distribution with mean λ, $\lambda^n \exp(-\lambda)/n!$, and binomial$(y; n, p)$ 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 numerical 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,

$$\text{Poisson}(n;\lambda) = \text{Poisson}(n-1;\lambda)\frac{\lambda}{n}$$

and

$$\text{binomial}(y;n,p) = \text{binomial}(y;n-1,p)\frac{n}{n-y}(1-p).$$

And then also for the Poisson-binomial product,

$$\text{Poisson}(n;\lambda)\ \text{binomial}(y;n,p) = \text{Poisson}(n-1;\lambda)\ \text{binomial}(y;n-1,p)\frac{\lambda}{n-y}(1-p).$$

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

$$\begin{aligned}
\text{Prob}(y) &= \sum_{n=y}^{n_{\max}} \text{Poisson}(n;\lambda)\ \text{binomial}(y;n,p) \\
&= \text{Poisson}(y;\lambda)\ \text{binomial}(y;y,p)\Big\{1 + f_1 + f_1 f_2 + \ldots + f_1 \cdots f_{n_{\max}}\Big\} \\
&= \text{Poisson}(y;\lambda)\ \text{binomial}(y;y,p)\Big\{1 + f_1(1 + f_2(1 + f_3(1 + \ldots)))\Big\}.
\end{aligned}$$

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

```
fac = 1; ff = lambda * (1-p)
for (i in (n.max - y):1) fac = 1 + fac * ff / i
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 chose the upper limit $n_{\max}$ up-front, 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 numerical stable.

7. References

Chandler, R. B., J. A. Royle, and D. I. King. 2011. Inference about density and temporary emigration in unmarked populations. Ecology 92:1429–1435.

Dail, D., and L. Madsen. 2011. Models for estimating abundance from repeated counts of an open metapopulation. Biometrics 67:577–587.

Denes, F. V., L. F. Silveira, and S. R. Beissinger. 2015. Estimating abundance of unmarked animal populations: accounting for imperfect detection and other sources of zero inflation. Methods in Ecology and Evolution 6:543–556.

Denwood, M. J. 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.

Dodd, C. K., and R. M. Dorazio. 2004. Using counts to simultaneously estimate abundance and detection probabilities in a salamander community. Herpetologica 60:468–478.

Fiske, I., R. Chandler, and others. 2011. unmarked: An R package for fitting hierarchical models of wildlife occurrence and abundance. Journal of Statistical Software 43:1–23.

Gong, L., and J. M. Flegal. 2016. A practical sequential stopping rule for high-dimensional Markov chain Monte Carlo. Journal of Computational and Graphical Statistics 25:684–700.

Joseph, L. N., C. Elkin, T. G. Martin, and H. P. Possingham. 2009. Modeling abundance using N-mixture models: the importance of considering ecological mechanisms. Ecological Applications 19:631–642.

Kéry, M. 2010. Introduction to WinBUGS for ecologists: Bayesian approach to regression, ANOVA, mixed models and related analyses. Academic Press.

Kéry, M., R. M. Dorazio, L. Soldaat, A. Van Strien, A. Zuiderwijk, and J. A. Royle. 2009. Trend estimation in populations with imperfect detection. Journal of Applied Ecology 46:1163–1172.

Kéry, M., and J. A. Royle. 2010. Hierarchical modelling and estimation of abundance and population trends in metapopulation designs. Journal of Animal Ecology 79:453–461.

Kéry, M., and J. A. Royle. 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.

Kéry, M., J. A. Royle, and H. Schmid. 2005. Modeling avian abundance from replicated counts using binomial mixture models. Ecological applications 15:1450–1461.

Kéry, M., and M. Schaub. 2011. Bayesian population analysis using WinBUGS: a hierarchical perspective. Academic Press.

Lunn, D., C. Jackson, N. Best, A. Thomas, and D. Spiegelhalter. 2012. The BUGS book: a practical introduction to Bayesian analysis. CRC press, Boca Raton, Florida.

Martin, J., J. A. Royle, D. I. Mackenzie, H. H. Edwards, M. Kéry, and B. Gardner. 2011. Accounting for non-independent detection when estimating abundance of organisms with a Bayesian approach. Methods in Ecology and Evolution 2:595–601.

O'Donnell, K. M., F. R. Thompson III, and R. D. Semlitsch. 2015. Partitioning detectability components in populations subject to within-season temporary emigration using binomial mixture models. PLoS ONE 10:e0117216.

Pollock, K. H., J. D. Nichols, T. R. Simons, G. L. Farnsworth, L. L. Bailey, and J. R. Sauer. 2002. Large scale wildlife monitoring studies: statistical methods for design and analysis. Environmetrics 13:105–119.

R Core Team. 2016. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Royle, J. A. 2004. N-mixture models for estimating population size from spatially replicated counts. Biometrics 60:108–115.

Royle, J. A., and R. M. Dorazio. 2008. Hierarchical modeling and inference in ecology: the analysis of data from populations, metapopulations and communities. Academic Press.

Royle, J. A., and J. D. Nichols. 2003. Estimating abundance from repeated presence–absence data or point counts. Ecology 84:777–790.

Rue, H., S. Martino, and N. Chopin. 2009. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 71:319–392.

Rue, H., S. Martino, F. Lindgren, D. Simpson, and A. Riebler. 2013. R-INLA: approximate Bayesian inference using integrated nested Laplace approximations. Trondheim, Norway.

Wenger, S. J., and M. C. Freeman. 2008. Estimating species occurrence, abundance, and detection probability using zero-inflated distributions. Ecology 89:2953–2959.

Yoccoz, N. G., J. D. Nichols, and T. Boulinier. 2001. Monitoring of biological diversity in space and time. Trends in Ecology & Evolution 16:446–453.