

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

Timothy D. Meehan and Håvard Rue†*

March 21, 2017

1. Introduction

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 the secretive nature of many wildlife species. 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” (Kery and Royle 2010), involves repeat visits in rapid succession to each of multiple study sites in a study area. If abundance at a site is assumed to be constant across repeat visits, an assumption referred to as “population closure”, 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 or binomial mixture model (Royle and Nichols 2003, Dodd and Dorazio 2004, Royle 2004, Kery et al. 2005). A simple form of an N-mixture model describes observed counts 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,

$$N_i \sim \text{Poisson}(\lambda) \quad \text{and} \quad Y_{i,j}|N_i \sim \text{Binomial}(N_i, p).$$

λ is commonly modeled as a log-linear function of site-level covariates, as $\log(\lambda_i) = \beta_0 + \beta_1 * x_i$. Similarly, p is commonly modeled as $\text{logit}(p) = \alpha_0 + \alpha_1 * x_i$, a logit-linear function of site-by-survey covariates.

These estimation approaches can be extended to cover k distinct breeding seasons, which correspond with distinct years for annually-breeding wildlife species (Kery et al. 2009). In this case, population closure is assumed across j surveys within year k , but is relaxed across years (Kery et al. 2009). A simple specification of a multiple year model is $N_{i,k} \sim \text{Poisson}(\lambda_{i,k})$, $Y_{i,j,k}|N_{i,k} \sim \text{Binomial}(N_{i,k}, p_{i,k})$. There are other variations of N-mixture models that accommodate overdispersed counts through use of a negative binomial distribution (Kery and Royle 2010), zero-inflated Poisson distribution (Wenger and Freeman 2008), or observation-level random intercepts (Kery 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. Under the right conditions, abundance estimates can be considerably more accurate than those that ignore detection error. 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

*National Audubon Society, Boulder, Colorado USA

†King Abdulla University of Science and Technology, Thuwal SAU

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 learning resources aimed at teaching this skill (Royle and Dorazio 2008, Kery 2010, Kery and Schaub 2011, Kery 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. Coding, tuning, and troubleshooting BUGS models could be considered as much an art as a science.

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, 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 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 rich inferential framework associated with Bayesian analysis (although this capacity is being added in increments).

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). 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 when compared to MCMC approaches. The R-INLA approach is different from unmarked in that inference about model parameters falls within a Bayesian paradigm.

The purpose of this manuscript is to demonstrate analysis of N-mixture models using the R-INLA package. In the process, we simulate a realistic dataset, and analyze it in a Bayesian context using both JAGS, via the runjags package (Denwood 2016) for R, and the R-INLA package. In each case, we demonstrate how data is prepared, how models are specified, how model estimates compare to simulation inputs, and how methods compare in terms of computational performance.

2. Data simulation

2.1 Hypothetical scenario

The data simulated for this analysis was 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 75 study sites. At each site, 4 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 (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 (covariates 1 and 3). The first covariate for detection, covariate 1, is the same site level covariate 1 that affects abundance, although it has the opposite effect on detection. The other detection covariate, covariate 3, is a site-year variable that could be related to average weather conditions during the observations at a given site and year.

Finally, as is commonly the case, we assume that the counts are overdispersed due to the effects of unknown variables. Overdispersion is accounted for using a negative binomial distribution for the count component of the model. The specific model parameters and their simulated values are: $p.b0 = 1.0$ (detection intercept), $p.b1 = -2.0$ (effect of detection covariate 1), $p.b3 = 0.5$ (effect of detection covariate 3), $lam.b0 = 2.0$

(abundance intercept), $\text{lam.b1} = 2.0$ (effect of abundance covariate 1), $\text{lam.b2} = -3.0$ (effect of abundance covariate 2), $\text{lam.b4} = 1.0$ (effect of year on abundance), $\text{disp.size} = 3.0$ (size of the overdispersion parameter). All independent variables in the simulation are centered at zero to alleviate computational difficulties and to make model intercepts more easily interpreted.

2.2 Simulation function

We simulated data for this study using the following function.

```
data4sim <- function(n.sites = 75,      # number study sites
                    n.surveys = 4,      # 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 overdispersion
){
  # setup
  if(n.years %% 2 == 0) {n.years <- n.years + 1} # make years odd
  y <- array(dim = c(n.sites, n.surveys, n.years)) # array for counts
  N.tr <- array(dim = c(n.sites, n.years)) # array for true abund
  # abundance covariates
  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 2
  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] <- rnbino(n = 1, mu = lam.tr[i, k], size = disp.size) # true N
    }
  }
  # detection covariates
  x.p.1 <- array(x.lam.1[,1], dim = c(n.sites, n.surveys, n.years))
  x.p.3 <- array(as.numeric(scale(rnorm(n = n.sites * n.years, 0, 0.25),
                                     scale = F)), dim = c(n.sites, n.years))
  # expand x.p.3
  out1 <- c()
  for(k in 1:n.years){
    chunk1 <- x.p.3[,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
  p.tr <- 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){
      y[i, j, k] <- rbinom(1, size = N.tr[i, k], prob = p.tr[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.lam.1 = x.lam.1[,1], x.lam.2 = x.lam.2[,1], yr = yr[1, ],
             y = y, lam.tr = lam.tr, N.tr = N.tr))
} #end

sim.data <- data4sim()

```

3.0 Data analysis using JAGS

3.1 JAGS model specification

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])

    } # close i loop
  } # close k loop

  # 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
"

```

3.2 JAGS parameters, dataset, and initial values

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$y, x.lam.1 = sim.data$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,
                 nSites = sim.data$n.sites, nSurveys = sim.data$n.surveys,
                 nYears = sim.data$n.years)

# initial values
N.init <- sim.data$y # 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))

# view data
str(jags.data, digits.d = 1)
```

```
## List of 9
## $ y      : int [1:75, 1:4, 1:9] 1 5 3 0 0 3 0 0 1 4 ...
## $ x.lam.1 : num [1:75] 0.34 0.3 -0.02 -0.08 -0.01 ...
## $ x.lam.2 : num [1:75] 0.47 0.46 0.09 0.47 0.31 ...
## $ yr      : num [1:9] -0.5 -0.4 -0.2 -0.1 0 ...
## $ x.p.1   : num [1:75] 0.34 0.3 -0.02 -0.08 -0.01 ...
## $ x.p.3   : num [1:75, 1:9] 0.06 0.22 -0.34 0.23 0.33 ...
## $ nSites  : num 75
## $ nSurveys: num 4
## $ nYears  : num 9
```

3.3 Run JAGS model via runjags and view results

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 shorter than necessary.

```
# set run parameters
nc <- 3; na <- 100; nb <- 100; ni <- 500; nt <- 1

# run jags
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")
```

```
# view results
round(as.data.frame(summary(out.jags)[,c(2, 1, 3)]), 2)
```

```
##           Median Lower95 Upper95
## intP      1.00    0.91    1.09
## bCov1P    -2.13   -2.43   -1.88
## bCov3P     1.15    0.84    1.45
## intLam     2.02    1.96    2.09
## bCov1Lam   1.96    1.76    2.16
## bCov2Lam  -2.95   -3.15   -2.75
## bYr        1.04    0.87    1.20
## overDisEst 2.96    2.52    3.42
```

Mean parameter estimates from the JAGS model are reasonably close to, and not significantly different from, the input values used to generate the data: detection intercept = 0.97 versus 1.0, effect of detection covariate 1 = -1.86 versus -2.0, effect of detection covariate 3 = 1.11 versus 1.00, abundance intercept = 2.04 versus 2.0, effect of abundance covariate 1 = 2.09 versus 2.0, effect of abundance covariate 2 = -2.93 versus -3.0, effect of year on abundance = 0.97 versus 1.0, and size of the overdispersion parameter = 2.90 versus 3.0. The full marginal posterior distributions for model parameters are shown in Figure 1. The potential scale reduction factor for all variables was < 1.05 , and the effective sample size for all variables was > 1640 . The simulation ran in parallel on 3 virtual cores, 1 MCMC chain per core, and took approximately 20 minutes.

4.0 Analysis using R-INLA

4.1 Prepare data for R-INLA

Next, we prepare data for R-INLA, which involved slight modification of the data in the `sim.data` object, which produced data formatted for JAGS. 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 lambda. 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`.

```
# format count data
y.inla <- sim.data$y[ , , 1]
for(i in 2:sim.data$n.years){
  y.chunk <- sim.data$y[ , , 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)

# make inla.mdata object
counts.and.count.covs <- inla.mdata(y.inla, 1, x.lam.1.inla, x.lam.2.inla, yr.inla)
str(counts.and.count.covs, digits.d = 2)
```

```
## List of 8
## $ Y1: int [1:675] 1 5 3 0 0 3 0 0 1 4 ...
## $ Y2: int [1:675] 1 6 3 0 0 3 0 0 1 4 ...
## $ Y3: int [1:675] 1 4 3 0 0 3 0 0 1 3 ...
```

```
## $ Y4: int [1:675] 1 3 4 0 0 3 0 0 0 3 ...
## $ X1: num [1:675] 1 1 1 1 1 1 1 1 1 1 ...
## $ X2: num [1:675] 0.344 0.299 -0.024 -0.083 -0.011 ...
## $ X3: num [1:675] 0.469 0.457 0.086 0.472 0.306 ...
## $ X4: num [1:675] -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 -0.5 ...
## - attr(*, "class")= chr "inla.mdata"
```

4.2 Analyze model using R-INLA and view results

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 covariates for p . 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 for the count component of the model. The fourth and fifth arguments specify the priors for the two model components, which are similar to those in the JAGS model.

```
# run inla model
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))))))

# view output
round(out.inla$summary.fixed[,c(4, 3, 5)], 2)
```

```
##              0.5quant 0.025quant 0.975quant
## (Intercept)      1.00      0.91      1.08
## x.p.1.inla       -2.16     -2.46     -1.86
## x.p.3.inla        1.14      0.84      1.44
```

```
round(out.inla$summary.hyperpar[,c(4, 3, 5)], 2)
```

```
##              0.5quant 0.025quant 0.975quant
## beta[1] for NMix observations      2.02      1.96      2.09
## beta[2] for NMix observations      1.97      1.76      2.18
## beta[3] for NMix observations     -2.94     -3.13     -2.75
## beta[4] for NMix observations      1.04      0.88      1.21
## overdispersion for NMix observations  0.33      0.28      0.39
```

Mean parameter estimates from the R-INLA model are also close to, and not significantly different from, input values: detection intercept = 0.97 versus 1.0, effect of detection covariate 1 = -1.86 versus -2.0, effect of detection covariate 3 = 1.11 versus 1.0, abundance intercept = 2.04 versus 2.0, effect of abundance covariate 1 = 2.09 versus 2.0, effect of abundance covariate 2 = -2.93 versus -3.0, effect of year on abundance = 0.97 versus 1.0, and size of the overdispersion parameter = $1 / 0.34 = 2.94$ versus 3.0. Full marginal posterior distributions from R-INLA are also shown in Figure 1. The analysis took approximately 6 seconds.

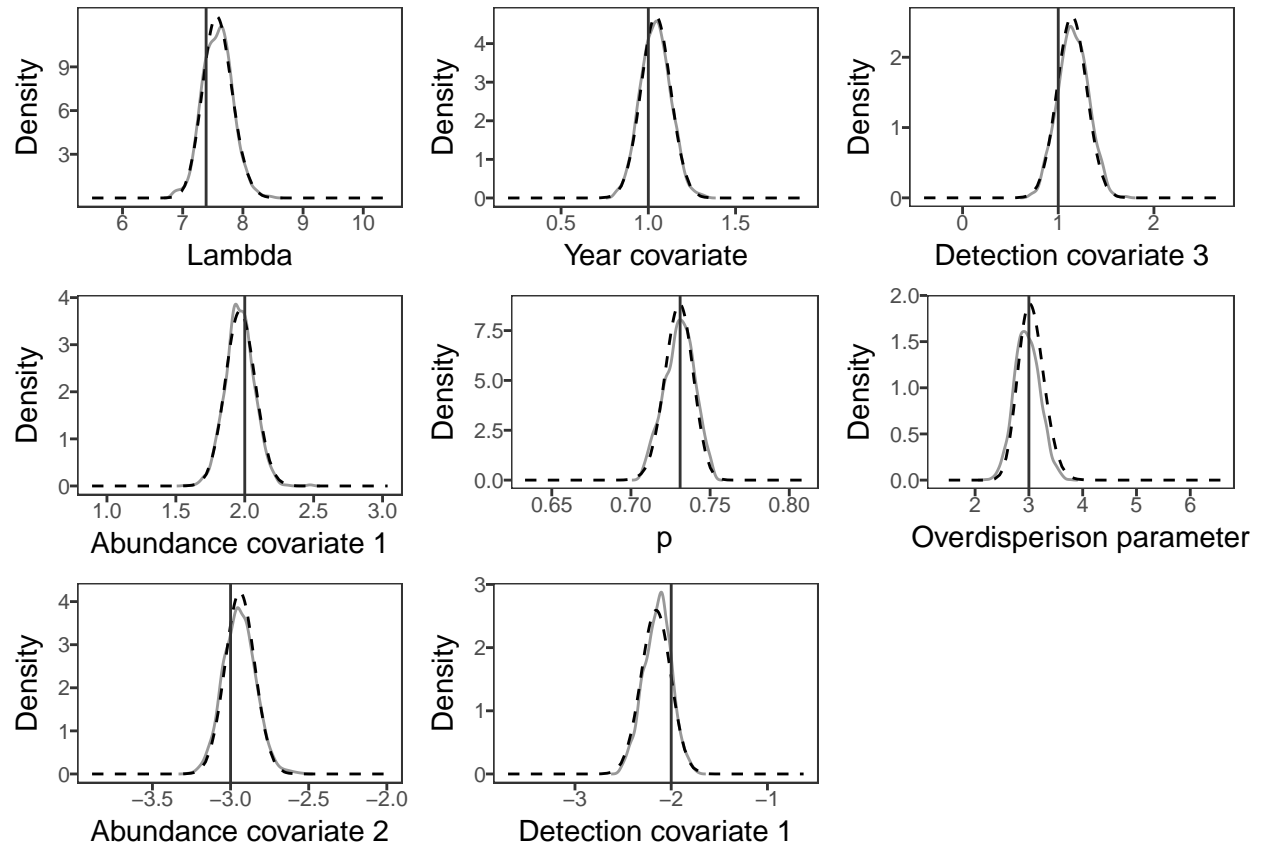


Figure 1: Figure 1. Marginal posteriors from JAGS (solid gray lines) and R-INLA (dashed black lines).

5.0 Discussion

The purpose of this study was to (1) demonstrate the use of the R-INLA package to analyze N-mixture models used by wildlife biologists and (2) compare performance of R-INLA to another common approach, JAGS via the runjags package, which uses MCMC methods.

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 lambda are specified through an R-INLA object included on the left side of the formula.

Regarding performance, both R-INLA and JAGS successfully extracted simulation input values. Figure 1 depicts marginal posterior distributions produced by JAGS and R-INLA. These posteriors 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, which is what occurs 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. Where R-INLA and JAGS differed was in computing time. In this example, MCMC took 1,215 seconds to produce posteriors while INLA took 6 seconds – a 200-fold difference. This was the case despite the fact that 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, 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.

There are also other differences between the two approaches that, in some cases, will favor the use of JAGS and other BUGS oriented approaches, despite their long processing times. BUGS based approaches allow users ultimate flexibility in specifying models, whereas, when using R-INLA, only a subset of possible models can be specified. For example, when using R-INLA, random effects cannot be specified for λ . However, it is possible to specify a wide variety of random effects for p , including simple exchangeable intercepts as well as spatially- and temporally-structured random effects. As another example, when using R-INLA, covariates for λ and p need to have the same degree of granularity, e.g., it is not possible to have site-year covariates for lambda and site-survey-year covariates for p . This limitation might become important if survey level covariates are not somewhat controlled by the sampling design. In this case, using the means of survey-level covariates per site and year will, at least, allows users to correct for any systematic differences in site-survey-year covariates across sites that might confound estimates of other site-level effects.

6.0 Appendix A

6.1 Likelihood

Would you like to add stuff here about the likelihood functions and the iterative procedure for solving them?

7.0 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.
- Kery, M. 2010. Introduction to WinBUGS for ecologists: Bayesian approach to regression, ANOVA, mixed models and related analyses. Academic Press.
- Kery, 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.
- Kery, 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.
- Kery, 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.
- Kery, M., J. A. Royle, and H. Schmid. 2005. Modeling avian abundance from replicated counts using binomial mixture models. *Ecological applications* 15:1450–1461.
- Kery, 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.