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

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

*March 31, 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 methodological 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" (Kery 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 , emmigration, 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 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 $Y$ 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 ($\lambda$). Specifically,

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

$\lambda$ 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 $logit(p_{i,j}) = \alpha_0 + \alpha_1 * x_{i,j}$, 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 $N_{i,k} \sim Poisson(\lambda_{i,k})$, $\quad Y_{i,j,k}|N_{i,k} \sim Binomial(N_{i,k}, p_{i,k})$. Like the single year specification, $\lambda$ is commonly modelled using site and site-by-year covariates, and $p$ using site-by-survey-by-year covariates. 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.

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

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 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 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 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 (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). INLA was developed specifically 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 Objectives

The purpose of this manuscript is to demonstrate analysis of N-mixture models using the R-INLA package. In the process, we explore simulated and actual count datasets, and analyze them using JAGS, via the runjags package (Denwood 2016) for R, the unmarked package, and the R-INLA package. 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 will also explore a limitation associated with the 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-by-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 abundance estimates under certain conditions. Most of the code used to conduct analyses is shown in body of this manuscript. However, some code related to generating tables and figures is not shown, for brevity, although it can be accessed via https://github.com/tmeeha/inlaNMix.

### 1.3 Simulated data

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 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 (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-survey-year variable that could be related to weather conditions during a given count.

Finally, as is commonly the case, we assume that the counts are overdispersed due to the effects of unknown variables. Overdispersion is generated 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 = 1.0 (effect of detection covariate 3), lam.b0 = 2.0 (abundance intercept), lam.b1 = 2.0 (effect of abundance covariate 1), lam.b2 = -3.0 (effect of abundance covariate 2), lam.b4 = 1.0 (effect of year on abundance), 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.

We simulated data for this study using the following function. Note that the function produces two versions of one detection covariate, 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 excersize 2, below. 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 here, in exercize 1.

```
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
```

```r
  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){
    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 exercise 3. This dataset comes from a study by Kery et al. (2005) and is publically available as part of the unmarked package. The dataset includes mallard duck counts, conducted at 239 sites on 2 or 3 occasions during the summer of 2002 as part of a Swiss program that monitors the abundance of breeding birds. In addition to counts, the dataset also includes 2 site-by-survey covariates related to detection (effort and survey date), and 3 site-level covariates related to abundance (route length, elevation, and forest cover). Full dataset details are given in Kery et al. (2002).

## 2. Exercise 1

### 2.1 Goals

In Excersize 1, we demonstrate the use of R-INLA and compare use and performance to a simular analyses using JAGS and unmarked. In this excercise, the forms of JAGS, unmarked, and INLA models match the data generating process. Specifically, we used x.p.3.mean to generate y1 and analyzed the data using models that use x.p.3.mean as a covariate. This exercise demonstrates the differences in package use, computation time, and estimation results.

### 2.2 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 $\lambda$ 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
"
```

### 2.3 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$y1, x.lam.1 = sim.data$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$n.years)
# initial values
N.init <- sim.data$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))
```

### 2.4 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 <- 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")
```

```
# view summary
summary(out.jags)
```

```
##               Lower95     Median  Upper95       Mean         SD Mode
## intP         0.855750  0.9726750  1.08688  0.9723813 0.05954747   NA
## bCov1P      -2.670200 -2.2615550 -1.84798 -2.2617876 0.20835586   NA
## bCov3P       0.235917  0.8082340  1.36188  0.8090189 0.28638687   NA
## intLam       1.921480  1.9938950  2.06540  1.9938209 0.03701274   NA
## bCov1Lam     1.969160  2.1975750  2.44411  2.1989919 0.12098830   NA
## bCov2Lam    -3.449230 -3.2482800 -3.03073 -3.2482299 0.10741855   NA
## bYr          0.795452  0.9682595  1.14234  0.9683030 0.08874730   NA
## overDisEst   2.393370  2.8182050  3.30055  2.8282076 0.23428035   NA
##                   MCerr MC%ofSD SSeff       AC.100     psrf
## intP       0.0012499167     2.1  2270  0.075001753 1.002574
## bCov1P     0.0045287157     2.2  2117  0.080299425 1.000450
## bCov3P     0.0049161214     1.7  3394  0.010047131 1.000469
## intLam     0.0005668737     1.5  4263  0.038161414 1.000572
## bCov1Lam   0.0020342394     1.7  3537  0.060088483 1.000338
## bCov2Lam   0.0008840492     0.8 14764  0.000329172 0.999920
## bYr        0.0007206051     0.8 15168 -0.011343986 1.000004
## overDisEst 0.0019326844     0.8 14694  0.008559271 1.000186
```

```
# computing time
round(jags.time.1 <- proc.time() - ptm, 2)[3]
```

```
## elapsed
## 2162.42
```

Mean parameter estimates from the JAGS model are reasonably close to, and not significantly different from, the input values used to generate the data (Table 1). 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.

### 2.5 Prepare data for unmarked

Next, we prepare data for the unmarked package, which involved 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-by-year variable, x.p.3, 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.

```
# 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$n.years)
x.lam.2.unmk <- rep(sim.data$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)
```

### 2.6 Analyze model using unmarked and view results

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.

```
# 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")
```

```
# view summary
summary(out.unmk)
```

```
##
## Call:
## pcount(formula = ~x.p.1.unmk + x.p.3.unmk ~ x.lam.1.unmk + x.lam.2.unmk +
##     yr.unmk, data = unmk.data, mixture = "NB")
##
## Abundance (log-scale):
##              Estimate     SE     z   P(>|z|)
## (Intercept)     1.969 0.0329  59.8  0.00e+00
## x.lam.1.unmk    2.108 0.1062  19.9  1.02e-87
## x.lam.2.unmk   -3.243 0.1069 -30.3 3.12e-202
## yr.unmk         0.965 0.0879  11.0  4.51e-28
##
## Detection (logit-scale):
##             Estimate     SE      z  P(>|z|)
## (Intercept)    1.031 0.0499  20.64 1.24e-94
## x.p.1.unmk    -2.040 0.1598 -12.77 2.41e-37
## x.p.3.unmk     0.777 0.2805   2.77 5.60e-03
##
## Dispersion (log-scale):
##  Estimate    SE    z  P(>|z|)
##      1.05 0.082 12.8 1.51e-37
##
## AIC: 7835.545
## Number of sites: 648
## optim convergence code: 0
## optim iterations: 49
## Bootstrap iterations: 0
```

```
# computing time
round(unmk.time.1 <- proc.time() - ptm, 2)[3]
```

```
## elapsed
##   80.69
```

Maximum liklihood estimates from the unmarked analysis are are also close to, and not significantly different from, input values (Table 1) The analysis took approximately 120 seconds.

### 2.7 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$y1[ , , 1]
for(i in 2:sim.data$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)
```

### 2.8 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 $\lambda$. 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
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)))))
```

```
# view summary
summary(out.inla)
```

```
##
## Call:
## c("inla(formula = counts.and.count.covs ~ 1 + x.p.1.inla + x.p.3.inla, ",  "    family = \"nmixnb\",
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.5478          6.3865          0.3000          7.2342
##
## Fixed effects:
##                 mean     sd 0.025quant 0.5quant 0.975quant    mode kld
## (Intercept)   0.9749 0.0604     0.8544   0.9756     1.0917  0.9772   0
## x.p.1.inla  -2.2568 0.2053    -2.6618  -2.2561    -1.8558 -2.2542   0
## x.p.3.inla   0.8096 0.2838     0.2512   0.8099     1.3654  0.8107   0
##
## The model has no random effects
##
## Model hyperparameters:
```

```
##                                          mean     sd 0.025quant 0.5quant
## beta[1] for NMix observations          1.9923 0.0362     1.9213   1.9922
## beta[2] for NMix observations          2.1955 0.1199     1.9624   2.1945
## beta[3] for NMix observations         -3.2468 0.1068    -3.4582  -3.2462
## beta[4] for NMix observations          0.9663 0.0878     0.7925   0.9667
## overdispersion for NMix observations   0.3510 0.0287     0.2983   0.3498
##                                       0.975quant    mode
## beta[1] for NMix observations              2.063  1.9921
## beta[2] for NMix observations              2.433  2.1911
## beta[3] for NMix observations             -3.038 -3.2443
## beta[4] for NMix observations              1.138  0.9682
## overdispersion for NMix observations       0.411  0.3472
##
## Expected number of effective parameters(std dev): 3.001(1e-04)
## Number of equivalent replicates : 215.90
##
## Marginal log-Likelihood:  -3945.56
```

```
# computing time
round(unmk.time.1 <- proc.time() - ptm, 2)[3]
```

```
## elapsed
##    7.58
```

Mean parameter estimates from the R-INLA model are also close to, and not significantly different from, input values (Table 1) Full marginal posterior distributions from R-INLA are also shown in Figure 1. The analysis took approximately 6 seconds.

```
##
##
## Table: Parameter estimates from JAGS, unmarked, and R-INLA
##
##                     JAGS Median    JAGS Lower CrL    JAGS Upper CrL    UNMK Estimate    UNMK Lower CL
## ----------------  -------------  ----------------  ----------------  ---------------  ---------------
## P Int                  0.97              0.86              1.09             1.03             0.93
## P Cov 1               -2.26             -2.67             -1.85            -2.04            -2.35
## P Cov 3                0.81              0.24              1.36             0.78             0.23
## Lambda Int             1.99              1.92              2.06             1.97             1.90
## Lambda Cov 1           2.20              1.97              2.44             2.11             1.90
## Lambda Cov 2          -3.25             -3.45             -3.03            -3.24            -3.45
## Lambda Year            0.97              0.80              1.14             0.97             0.79
## Overdispersion         2.82              2.39              3.30             1.05             0.89
```

### 2.9 Notes on Exercise 1

Exercise 1 shows several things.

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.

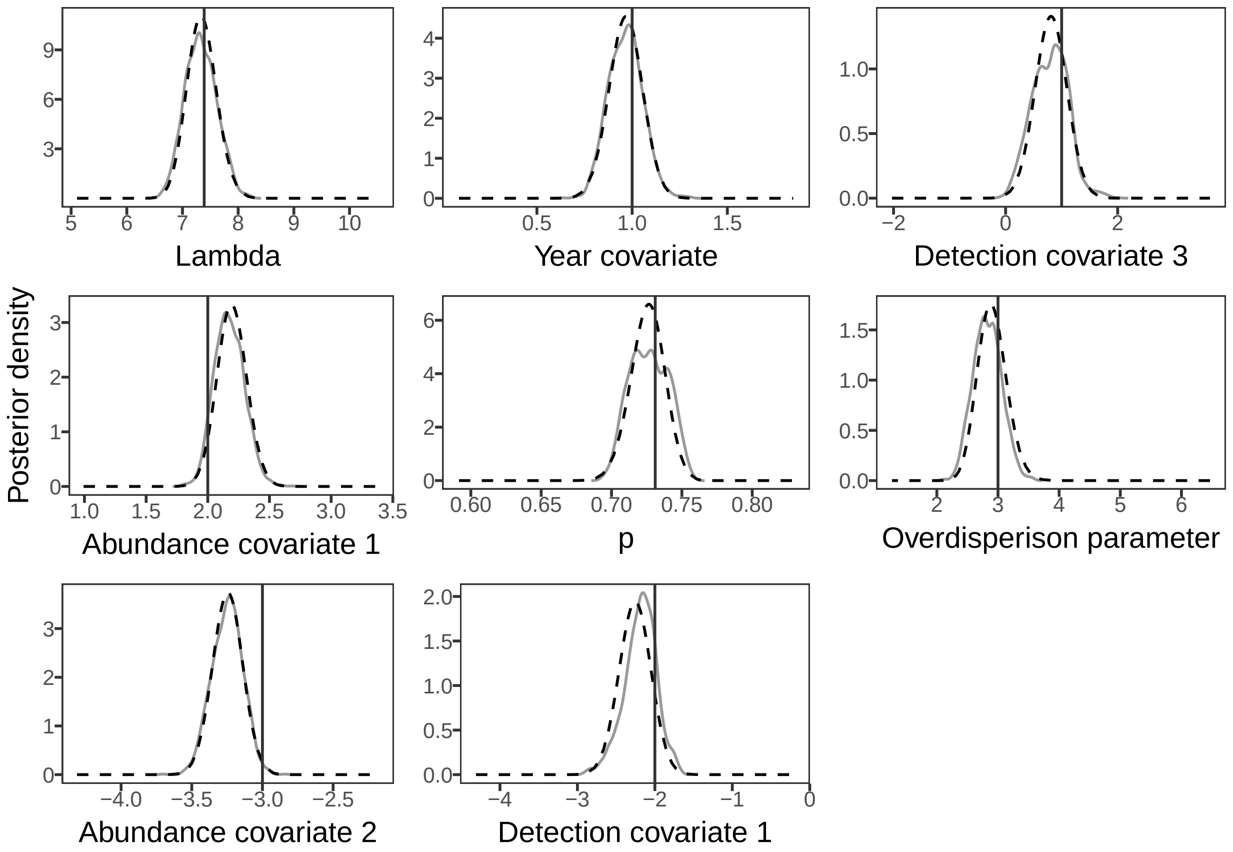


Figure 1: Figure 1. Marginal posteriors from JAGS (solid gray lines) and R-INLA (dashed black lines).

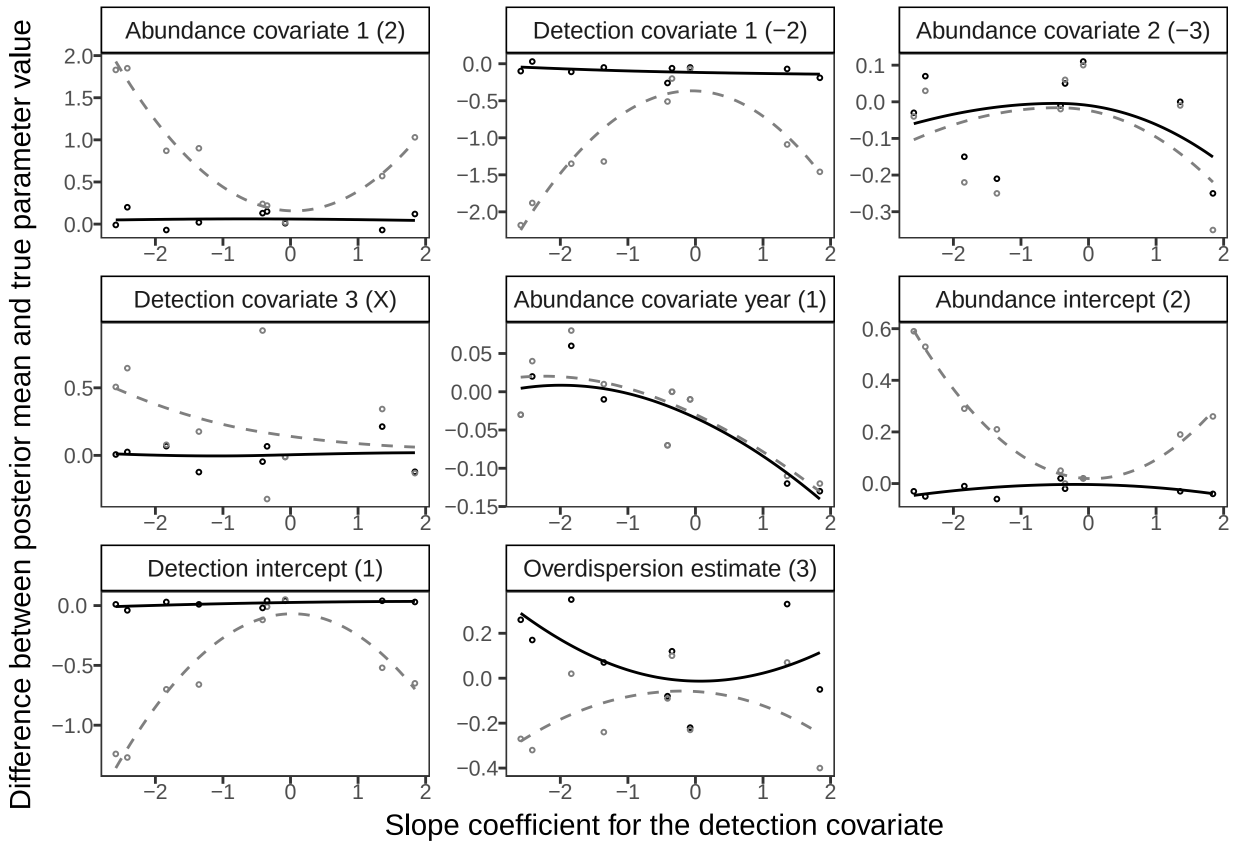
## 3. Example 2, Simulated Data

### 3.1 Goals

In Example 2, we show the consequences of not being able to specify a survey level covariate.

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 $\lambda$. 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 $\lambda$ 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.

```
## `geom_smooth()` using method = 'loess'
```



### 3.2 Results

This is what happened in this section.

## 4. Example 3, Real Data

### 4.1 Goals

We wanted to show real data.

### 4.1 unmarked

```
# get real mallard data from unmarked package
data(mallard)
# format unmarked data
mallard.y.unmk <- mallard.y
mallard.site.unmk <- mallard.site
```

```r
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)
```

```
## Warning: 4 sites have been discarded because of missing data.
```

```r
summary(mallard.out.unmk)
```

```
##
## Call:
## pcount(formula = ~ivel + date + I(date^2) ~ length + elev + forest,
##     data = mallardUMF, K = 30, mixture = "NB")
##
## 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) -0.02830 0.2846 -0.0994   0.921
## ivel         0.17419 0.2273  0.7663   0.444
## date        -0.31286 0.1467 -2.1322   0.033
## I(date^2)   -0.00474 0.0807 -0.0588   0.953
##
## Dispersion (log-scale):
##  Estimate    SE     z P(>|z|)
##    -0.695 0.364 -1.91  0.0558
##
## AIC: 477.5695
## Number of sites: 235
## Sites removed: 12 69 118 146
## optim convergence code: 0
## optim iterations: 79
## Bootstrap iterations: 0
```

### 4.2 INLA

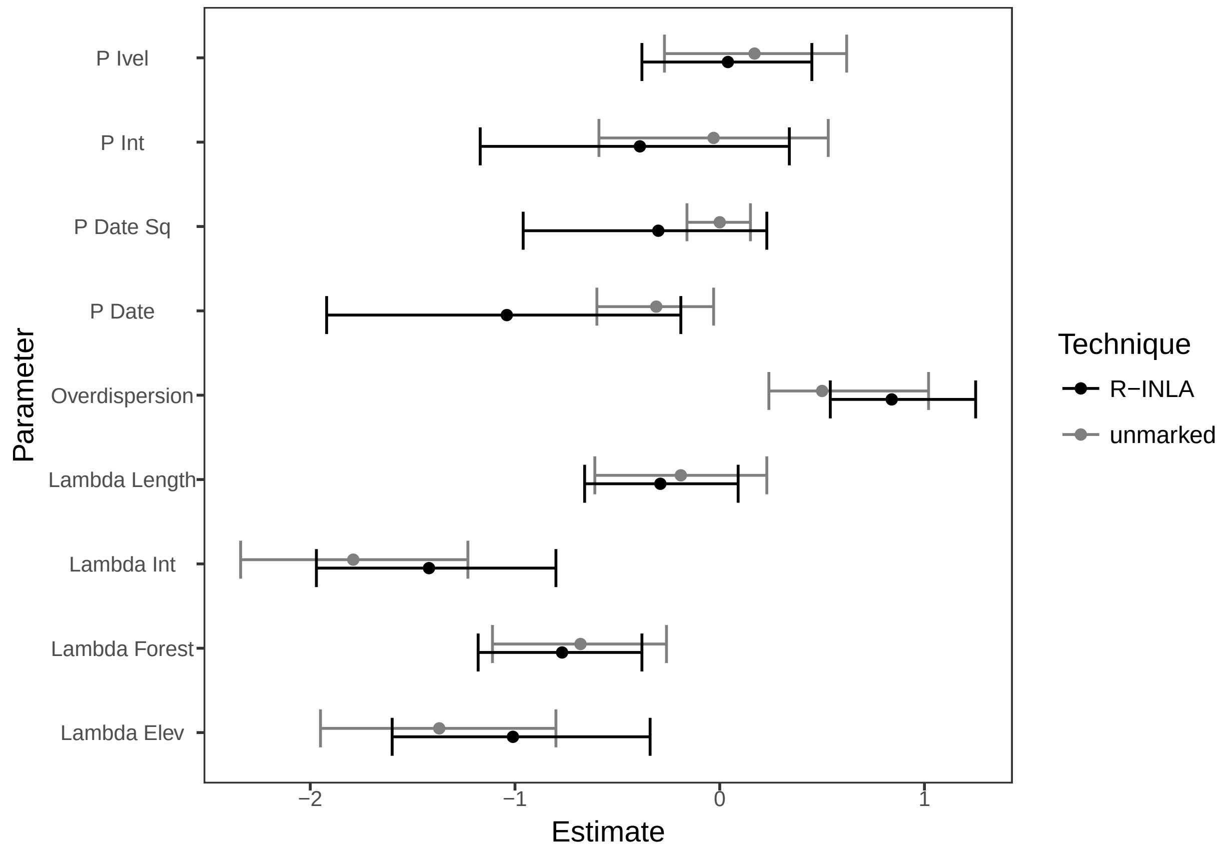
```r
# format inla data
mallard.y.inla <- mallard.y
mallard.length.inla <- mallard.site[,2]
mallard.elev.inla <- mallard.site[,1]
mallard.forest.inla <- mallard.site[,3]
mallard.ivel.inla <- rowMeans(mallard.obs$ivel, na.rm=T)
mallard.ivel.inla[is.na(mallard.ivel.inla)] <- mean(mallard.ivel.inla, na.rm=T)
mallard.date.inla <- rowMeans(mallard.obs$date, na.rm=T)
mallard.date2.inla <- mallard.date.inla^2
# make inla.mdata object
```

```r
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)))))
```

```r
summary(mallard.out.inla)
```

```
##
## Call:
## c("inla(formula = counts.and.count.covs ~ 1 + mallard.ivel.inla + ",  "    mallard.date.inla + malla
##
## Time used:
##  Pre-processing    Running inla Post-processing           Total
##          0.5023         12.9326          0.3156         13.7505
##
## Fixed effects:
##                       mean     sd 0.025quant 0.5quant 0.975quant    mode
## (Intercept)        -0.3969 0.3831    -1.1703  -0.3894     0.3351 -0.3738
## mallard.ivel.inla   0.0392 0.2115    -0.3778   0.0393     0.4549  0.0400
## mallard.date.inla  -1.0436 0.4330    -1.9229  -1.0358    -0.1945 -1.0158
## mallard.date2.inla -0.3179 0.3044    -0.9625  -0.3012     0.2334 -0.2680
##                    kld
## (Intercept)          0
## mallard.ivel.inla    0
## mallard.date.inla    0
## mallard.date2.inla   0
##
## The model has no random effects
##
## Model hyperparameters:
##                                          mean     sd 0.025quant 0.5quant
## beta[1] for NMix observations         -1.4118 0.2959    -1.9656  -1.4239
## beta[2] for NMix observations         -0.2904 0.1904    -0.6643  -0.2908
## beta[3] for NMix observations         -0.9984 0.3184    -1.5952  -1.0113
## beta[4] for NMix observations         -0.7706 0.2025    -1.1783  -0.7667
## overdispersion for NMix observations  1.2277 0.2642     0.7995   1.1946
##                                      0.975quant    mode
## beta[1] for NMix observations           -0.8014 -1.4674
## beta[2] for NMix observations            0.0856 -0.2923
## beta[3] for NMix observations           -0.3411 -1.0576
## beta[4] for NMix observations           -0.3818 -0.7527
## overdispersion for NMix observations     1.8376  1.1290
```

```
##
## Expected number of effective parameters(std dev): 3.997(4e-04)
## Number of equivalent replicates : 58.80
##
## Marginal log-Likelihood:  -270.91
```



This is what we found.

## 5. 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.

## 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.