CHAPTER

Integrated models®

20

Chapter outline

20.1 Introduction

Throughout this book, we have stressed the role of the generalized linear model (GLM) as a fundamental building block in parametric statistical modeling. Sometimes a modeling problem may be simple enough so that a GLM may be what you need. However, often we want to accommodate additional latent structure in the data and for this invoke latent variables or random effects, either in the simpler mixed models of Chapters 7, 10, 14, and 17 or in the form of a more general hierarchical model as for instance in Chapters 19 and 19B. Either type of hierarchical model can be viewed as the combination of two or more GLMs that are linked in sequence. That is, the latent response of one random process appears in the model for another response which may be latent or observed, and these subprocesses are organized according to conditional probability (Royle & Dorazio, 2008; Cressie et al., 2009; Hobbs & Hooten, 2015; Hooten & Hefley, 2019).

In this final modeling chapter in the book, we show another way in which we may combine multiple GLMs to build a more complex model. In this class of model, the component GLMs are not necessarily arrayed in sequence, as in a hierarchical model, but rather branch off from some shared underlying component. These are models that combine the information contained in multiple and disparate data sets. We call

®This book has a companion website hosting complementary materials, including all code for download. Visit this URL to access it: https://www.elsevier.com/books-and-journals/book-companion/9780443137150.

Applied Statistical Modelling for Ecologists. DOI: https://doi.org/10.1016/B978-0-443-13715-0.00001-7

them integrated models (IMs), because they integrate (or combine, fuse, assimilate) the information in multiple data sets. In contrast to a repeated-measures design as with an occupancy- or *N*-mixture model, in an IM there is always some degree of disparity in the different modeled data. That is, we have multiple data sets that each contain information about a shared process with one or more shared parameters. However, at first it may not be evident how to formally combine them in a single model. Our task then in developing an IM is (1) to identify the shared process and shared parameters and (2) write a model, or develop a likelihood, that links these shared parameters with the different types of observations in each data set.

Most of the time in an IM, one or more parameters in the latent state process are shared, and then we define different observation models for each data set. "Observation model" here must be understood in a broad sense. For instance, the underlying state may be population abundance, and the different data sets may be produced by a distance-sampling, removal-sampling, or capture-recapture protocol (Royle & Dorazio, 2008), at different spatial or temporal observation scales (requiring change-of-support modeling, Pacifici et al., 2019), or by various truncation, censoring, or aggregation processes.

IMs have become a mega-trend in ecological statistics during the past 30 years, e.g., in demographic estimation, population modeling, species distribution modeling, or movement modeling; for reviews, see Schaub & Abadi (2011), Maunder & Punt (2013), Zipkin & Saunders (2018), Miller et al. (2019), chapter 10 in Kéry & Royle (2021), and Schaub & Kéry (2022). The principles of IMs in these different subfields are very similar, but this is arguably not as widely understood as it should be, and IMs in each field are often viewed as something separate. Probably the most widely popularized kind of IM in ecology has been *integrated population models* (IPMs; Besbeas et al., 2002; Schaub & Kéry, 2022; Schaub et al., 2024). However, IPMs are really just a special case of IMs in general, which all share the same basic building principles.

Fig. 20.1 shows the typical case of an integrated species distribution model (SDM) (IM_1) and of an IPM (IM_2). In the former, we typically have a single shared parameter (θ), e.g., an abundance intensity in a point process model (Dorazio, 2014; Koshkina et al., 2017), expected abundance (Zipkin et al., 2017), or occupancy probability (Landau et al., 2022). The typical setting of an IPM is where we have a time series of population counts (data1), which contains information about all components of population dynamics including initial density (λ), survival (ϕ), and recruitment (γ), and then add another data set, for example, a capture–recapture data set, which contains only information about the survival (ϕ) part of population dynamics (Besbeas et al., 2002).

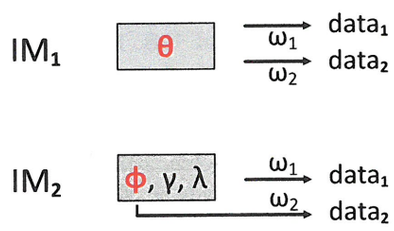

FIGURE 20.1

Schema of two kinds of integrated models where two disparate data sets inform about the same (or part of the same) underlying process (gray box) with shared parameter(s) shown in red (In IM_1, both data sets are informative about the same parameter(s) in the shared process, while in IM_2 data set 1 contains information about all three parameters, but data set 2 only about ϕ, which is the shared parameter. The arrows each denote a data-set-specific observation process, or likelihood, with parameters that are usually not shared (modified from Fig. 10.1 in Kéry & Royle, 2021).

In the context of SDMs, Pacifici et al. (2017) have discussed three different ways in which the information in different data sets may be combined and which they call the "covariate," "correlation," and "shared" approaches. The last one is what we here call an IM, and it consists in defining a joint likelihood for all parameters that govern the probability densities of all data sets. Under the simplifying assumption of statistical independence among the data sets, this joint likelihood is simply the product of the likelihoods of each individual data set, exactly analogous to how the likelihood of all the data points in a single data set is, under independence, just the product of the densities of each data point, as we have seen throughout the book. Maximizing the joint likelihood then identifies parameter values that are most likely for all data sets simultaneously.

Why should we build IMs? There are three good reasons for this: (1) It just makes tremendous sense to use all the information that is available in some estimation task. (2) More data sets can be viewed as akin to a larger sample size. Thus, it will not come as a surprise that parameter estimates in an IM are usually more precise than those obtained from each component data set individually. (3) Depending on the model and the kinds of data that are combined, parameters may become estimable in the IM which may not be identifiable from each data set alone (Schaub & Kéry, 2022).

But should we always build an IM when we have several data sets that we feel contain information about something shared? Our answer here would be, as so often, that it depends. Most of the times we would probably try to exploit the information of all data sets in an IM, but not always. For instance, in the context of an IPM, if you're most interested just in the population trends, then a simpler model applied only to the population count data may be more what you want. In contrast, if you're most interested in obtaining estimates of survival rates, then a Cormack-Jolly-Seber model (chapter 7 in Kéry & Schaub, 2012) applied to the capture–recapture data alone may be most appropriate for you. Building an IM incurs a cost: more parameters must be included in the model (typically in the observation models for each data set) and if the information about the target parameter is weak in an additional data set, then the inclusion of that data set in an IM may not be warranted.

Our goal in the rest of this chapter will be to examine the key features of an IM and to understand the principles of how we can build such a model. For this, we will develop an IM which is a very simple example of an SDM. We assume we have three data sets that contain information about spatial patterns in the distribution or abundance of common swifts (Fig. 20.2) in some region. Data set 1 comprises regular counts, but the other two data sets suffer from some loss of information relative to regular counts. In data set 2, we assume that the lazy bird-watchers had "forgotten" to record the nondetections and only recorded counts equal to 1 or greater. Such data are called zero truncated (ZT), since no zeroes occur in the data set. In data set 3, we only distinguish between a count of 0 (a nondetection) and a count of 1 or greater (i.e., a detection). In other words, these are what many people call "presence/absence data" and, statistically speaking, can also be called counts that are censored at 1. This chapter builds partly on Section 2.8 in Schaub & Kéry (2022).

Normally, our preference would be for an SDM that contains an explicit representation of the false-negative errors that always occur when collecting such data in the field. That is, for a type of occupancy or *N*-mixture model as shown in Chapters 19 and 19B. However, to introduce the ideas of IMs in a setting as simple as possible, we here ignore this complication of reality and assume that either detection probability is perfect or else that it does not covary with our comparison of interest, which will be an elevation gradient in abundance, and that we are happy with inferences about apparent or relative abundance.

FIGURE 20.2

A flock of common swifts (*Apus apus*) above the village of Glovelier, Swiss Jura mountains (Photo by Alain Georgy).

We will also make another assumption: that the three data sets we work with are statistically independent. Independence in IMs is a somewhat elusive concept. It has often been defined as meaning that no (or only few) individuals, or sites, are shared among the data sets to be combined in an IM (Besbeas et al., 2009; Abadi et al., 2010; Schaub & Kéry, 2022). Such "lack of sampling overlap" may often be a suitable indicator for statistical independence of the data sets, but may not be enough. Statistically speaking, independence is defined such that the joint density of some data sets is given by the product of the (joint) densities of the individual data sets (Blitzstein & Hwang, 2019). Independence in the context of an IM, the consequences of its violation, and the proper accommodation of nonindependence are areas of ongoing research (Besbeas et al., 2009; Abadi et al., 2010; Plard et al., 2021; Weegman et al., 2021; Schaub & Kéry, 2022; Barraquand et al., 2024). Here, we will assume that the three sets contain distinct sets of sites and thus will index sites with i, j, and k below. However, if there was some or even complete overlap among sites, then this could also be accommodated in an IM.

As a starting point, we might think that natural probability models for data sets 1 and 2 would be a Poisson GLM and a Bernoulli GLM with logit link (i.e., a logistic regression) for data set 3. Thus, in data set 1, for the count $y_i^{(1)}$ at site i, and where we indicate the data set by a superscript, we could write:

$$y_i^{(1)} \sim Poisson(\lambda_i)$$
$$\log(\lambda_i) = \alpha^{(1)} + \beta^{(1)} \cdot elev_i$$

Here, the elevation of site i affects the expected, or mean, count λ through a log-linear model with the two parameters $\alpha^{(1)}$ and $\beta^{(1)}$. Note that we like to simulate the effects of a covariate in our models in this chapter, but the data integration in this chapter would work just as well with a simpler intercept-only model.

As for data set 2, we might want to fit the same model also to the zero-truncated counts. But this would be wrong, since this would fail to account for the fact that zeros cannot be observed from the process that produced these data. Thus, the parameters estimated in such a naive application of a Poisson GLM to the zero-truncated data would not be comparable to the parameters estimated from the model for the "intact" data set 1. A proper probability model for the ZT Poisson data must recognize that zero is not in the sample space of the random experiment that produces data set 2. Thus, the proper probability mass function for the ZT data set 2 is a Poisson probability mass function (PMF) for counts >0 that we must renormalize using the sum of the probability of all possible outcomes, that is, of the nonzero counts only. Doing this ensures that the PMF sums to 1 over all possible outcomes, as required for a valid PMF. Once we do this, we can again go on to model the expected zero-truncated count with the same linear predictor as in the model for data set 1.

Let's look at this in algebra. The regular Poisson PMF is $p(y|\lambda) = \lambda^y e^{-\lambda}/y!$ (see Section 2.2.2.). The probability for a zero observation is $p(0|\lambda) = \lambda^0 e^{-\lambda}/0!$ which simplifies to $p(y=0|\lambda) = e^{-\lambda}$. Thus, the probability of a nonzero observation is $p(y>0|\lambda) = 1 - e^{-\lambda}$. Therefore, the PMF of the ZT Poisson is $p(y|\lambda) = \lambda^y e^{-\lambda}/((1-e^{-\lambda})y!)$. Below, we will implement this custom PMF for our do-it-yourself (DIY)-MLEs.

For data set 3, the binary detection/nondetection data $y_k^{(3)}$ at site k, we might write:

$$y_k^{(3)} \sim Bernoulli(\psi_k)$$
$$logit(\psi_k) = \alpha^{(3)} + \beta^{(3)} \cdot elev_k$$

Here, we model occurrence, or relative occupancy, probability on the logit scale as a linear function of site elevation, with parameters $\alpha^{(3)}$ and $\beta^{(3)}$.

Both the Poisson GLMs (with and without zero truncation) and the Bernoulli GLM offer a valid characterization of the spatial variation of the distribution of the study species: one in terms of relative abundance here denoted λ (Johnson, 2008) and the other in terms of relative occupancy probability here denoted ψ (Kéry, 2011). Note that the terms "relative" or "apparent" mean that true abundance or occurrence is confounded with imperfect detection, as described above. Intuitively, it seems obvious that patterns of occurrence also contain a signal of the underlying patterns of abundance. But would there be a possibility of combining the two models, for abundance and for occurrence, and thereby make use of all information about abundance that our data sets contain?

Key to data integration using a joint likelihood is always that you must describe the data sets using some "common currency." That is, with at least one parameter that is shared among two or more of the data sets combined. In our case, we can recognize that the detection/nondetection data are simply an information-poor variant of counts: they only distinguish between a count of 0 and one that is 1 or greater, or they are censored at 1. Thus, we can express the detection/nondetection data as a summary of an underlying abundance process. There are multiple ways in which such a link between abundance and the binary detection/nondetection data can be formulated (Royle & Dorazio, 2008). Here, we specify the linkage via that vastly underused link function for Bernoulli

or binomial data: the complementary log-log or cloglog link (Section 3.3.6 in Kéry & Royle, 2016; Scharf et al., 2022). As we will see, by specifying that link function for the apparent occurrence probability ψ in the model for the binary detection/nondetection data, we describe these binary data in terms of an underlying Poisson abundance process. Thus, the parameters in the linear model for ψ on the cloglog scale will describe the log-linear covariate relationship for the expectation of an assumed Poisson random variable that underlies the binary data.

Does this sound like magic? Well, we think it does a little ... but let's look at this with algebra. When we model the binary data as a summary of an underlying abundance or count, then the Bernoulli parameter is the probability to obtain a count greater than 0 under a Poisson abundance model. From the Poisson PMF, we can express this as

$$\psi = P(y=1) = P(N>0) = 1 - P(N=0) = 1 - \exp(-\lambda).$$

We could specify this relationship simply as a link function in a GLM, but for your understanding it is useful to derive the complementary log-log link by hand. Specification of a linear model for a Bernoulli parameter at the scale of a cloglog link transformation, that is, $\text{cloglog}(\psi) = \log(-\log(1-\psi))$, is equivalent to modeling linear effects in $\log(\lambda)$, which we can see by making the following re-arrangements:

$$\begin{aligned}
&\psi = 1 - \exp(-\lambda), \text{ and therefore} \\
&\exp(-\lambda) = 1 - \psi, \text{ therefore} \\
&\log(\exp(-\lambda)) = \log(1-\psi), \text{ therefore} \\
&-\lambda = \log(1-\psi), \text{ therefore} \\
&\lambda = -\log(1-\psi), \text{ and therefore we finally get this:} \\
&\log(\lambda) = \log(-\log(1-\psi)).
\end{aligned}$$

What we have on the right-hand side of the last line is the cloglog transformation for occupancy probability ψ.

Thus, for our three SDM data sets we have achieved the two typical tasks in integrated modeling: (1) we have identified abundance (λ) as a shared process underlying all three, and (2) we have chosen observation models for the three such that the response data in each are formally linked to this underlying, shared process.

We will simulate three data sets as just described by first simulating three sets of Poisson random variables. The first one will be left unchanged and will be our data set 1. In the second set, we will toss out all zeros and the remainder will be our data set 2. Finally, we quantize (or censor at 1) the third set resulting in detection/nondetection data; this will become our data set 3.

Then, we will do what is always a wise approach in an IM: fit the component models first to each data set alone. We will do this using canned functions (when available) in R, with a DIY likelihood function that we maximize with `optim()`, and with JAGS for Bayesian posterior inference. This will help us understand the full model, since to understand an IM, you must also understand all of its component models. In addition, since it is so easy to make coding mistakes, it is always good to build up more complex code from smaller sets of code that you think work correctly. Finally, strong changes in parameter estimates between the simple and the IM may indicate some problem.

Regardless of whether we fit an IM by maximum likelihood or by Bayesian posterior inference, we work with the same joint likelihood, which is the product of the likelihoods of each individual data set (under the important assumption of statistical independence!). The magic especially of BUGS-language software for IMs is that these models are trivially easy to specify: within the same model statement we

simply describe the likelihood for each data set and then define one or more shared parameters that have the same name and meaning in two or more of these component models. The wonderful ease of fitting even complex IMs in the BUGS language is the main reason for why Bayesian posterior inference has become the dominant mode of inference for IPMs (Schaub & Kéry, 2022). However, the concept of the joint likelihood as a product of the likelihoods of the individual data sets is buried somewhat in the Bayesian approach to fitting these models. In contrast, when doing likelihood inference, we will explicitly define the joint likelihood as such a product. Thus, it is conceptually valuable to see the IM fit in both ways. This is what we do in Section 20.4.

20.2 Data generation: simulating three abundance data sets with different observation/aggregation models

We start by simulating three count data sets with identical parameters for a log-linear regression of the expected counts on a covariate "elevation." Then, we degrade the information in data sets 2 (truncated Poisson) and 3 (detection/nondetection) by, respectively, discarding all zeroes and one-censoring the counts so they become binary detection/nondetections, or DND data. In real life, both of these "degraded" data types are logistically cheaper to obtain. Hence, we assume we have regular counts from 500 sites, zero-truncated counts from 1000 and detection/nondetection data from 2000. We sort the covariate values within each data set, which has no effect on the calculations, but makes plotting easier (Fig. 20.3).

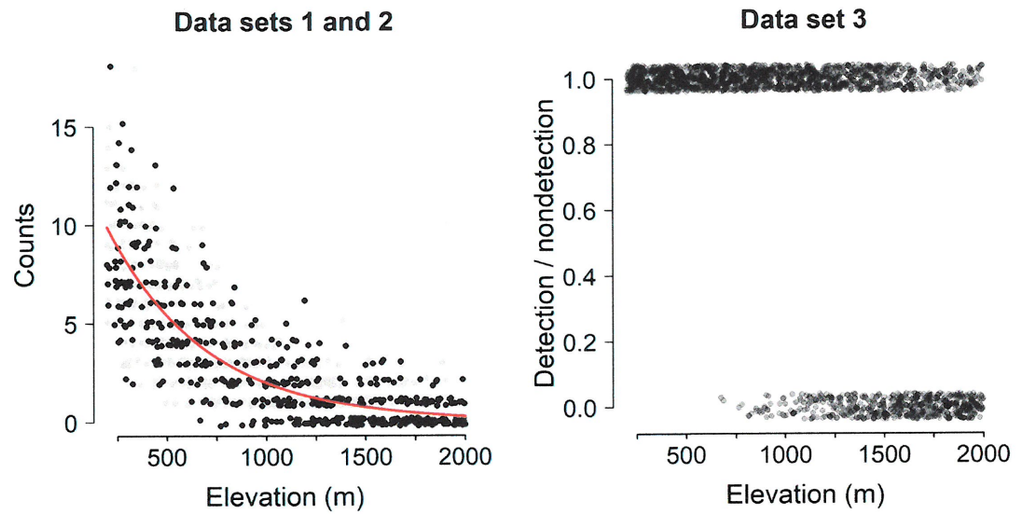

FIGURE 20.3

Simulated count (left) and detection/nondetection data (right) as a function of elevation. Original data are Poisson counts (left) with 500 sites in data set 1 (black) and 1000 sites minus 272 Poisson zeroes in the zero-truncated data set 2 (gray) and where the common expectation as a function of elevation is shown by the red curve. The detection/nondetection data (right) stem from 2000 sites. In the integrated model, we combine all three data sets in a single expression for the abundance-elevation relationship.

```
set.seed(20)

# Simulate the two data sets and plot them
# Choose sample size and parameter values for both data sets
nsites1 <- 500                              # Sample size for count data
nsites2 <- 1000                             # Sample size for zero-truncated counts
nsites3 <- 2000                             # Sample size for detection/nondetection data
mean.lam <- 2                               # Average expected abundance (lambda) per site
beta <- -2                                  # Coefficient of elevation covariate on lambda
truth <- c("log.lam" = log(mean.lam), "beta" = beta)  # Save truth

# Simulate elevation covariate for all three and standardize to mean of 1000 and
# standard deviation also of 1000 m
elev1 <- sort(runif(nsites1, 200, 2000))        # Imagine 200-2000 m a.s.l.
elev2 <- sort(runif(nsites2, 200, 2000))
elev3 <- sort(runif(nsites3, 200, 2000))
selev1 <- (elev1 - 1000)/1000                   # Scaled elev1
selev2 <- (elev2 - 1000)/1000                   # Scaled elev2
selev3 <- (elev3 - 1000)/1000                   # Scaled elev3

# Create three regular count data sets with log-linear effects
C1 <- rpois(nsites1, exp(log(mean.lam) + beta * selev1))
C2 <- rpois(nsites2, exp(log(mean.lam) + beta * selev2))
C3 <- rpois(nsites3, exp(log(mean.lam) + beta * selev3))

table(C1)   # Tabulate data set 1

C1
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  18
150  97  62  38  39  29  21  22  11  11   6   3   5   2   2   1   1

# Create data set 2 (C2) by zero-truncating (discard all zeroes)
ztC2 <- C2                  # Make a copy
ztC2 <- ztC2[ztC2 > 0]      # Tossing out zeroes yields zero-truncated data
table(C2); table(ztC2)      # tabulate both original and ZT data set

C2
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  18
272 204 136 105  59  56  43  35  36  15  16  10   6   4   1   1   1
ztC2
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  18
204 136 105  59  56  43  35  36  15  16  10   6   4   1   1   1

# Turn count data set 3 (C3) into detection/nondetection data (y)
y <- C3                     # Make a copy
y[y > 1] <- 1               # Squash to binary
table(C3)   ;   table(y)    # tabulate both original counts and DND

C3
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
528 428 288 175 140 101  70  68  51  56  36  28  14   8   4   2   2   1
Y
  0   1
528 1472
```



```
# Plot counts/DND data in all data sets (Fig. 20.3)
library(scales)
par(mfrow = c(1, 2), mar = c(5, 5, 4, 1), cex = 1.2, cex.lab = 1.5, cex.axis = 1.5, las = 1)
plot(elev2[C2>0], jitter(ztC2), pch = 16, xlab = 'Elevation (m)', ylab = 'Counts',
  frame = FALSE, ylim = range(c(C1, ztC2)), col = alpha('grey80', 1), main = 'Data sets 1 and 2')
points(elev1, jitter(C1), pch = 16)
lines(200:2000, exp(log(2) -2 * ((200:2000)-1000)/1000), col = 'red', lty = 1, lwd = 2)
axis(1, at = c(250, 750, 1250, 1750), tcl = -0.25, labels = NA)
plot(elev3, jitter(y, amount = 0.04), xlab = 'Elevation (m)', ylab = 'Detection/nondetection',
  axes = FALSE, pch = 16, col = alpha('grey60', 0.3), main = 'Data set 3')
axis(1)
axis(1, at = c(250, 750, 1250, 1750), tcl = -0.25, labels = NA)
axis(2, at = c(0, 1), labels = c(0, 1))

# Required libraries
library(ASMbook); library(jagsUI); library(rstan); library(TMB)
```

20.3 Fitting models to the three individual data sets first

In an IM, we will virtually always fit the component models to each individual data set first. We do this here, obtaining both DIY-MLEs in R and Bayesian posterior inference in JAGS. We also use canned functions in R where available.

20.3.1 Fitting a standard Poisson GLM in R and JAGS to data set 1

We start by fitting a Poisson GLM to data set 1 to see how well we can recover the two parameters of the abundance model. We do this with the iteratively reweighted least-squares method in R, which yields the MLEs for these models. We find estimates that are pretty close to the known truth.

```
# Get MLEs for individual data sets
# Data set 1: Poisson GLM with log link for counts
summary(fm1 <- glm(C1 ~ selev1, family = poisson(link = "log")))
exp(coef(fm1)[1])   # Estimate of lambda on natural scale from counts

Coefficients:
            Estimate   Std. Error   z value   Pr(>|z|)
(Intercept)  0.69493      0.03668     18.94     <2e-16 ***
selev1      -1.96130      0.06700    -29.27     <2e-16 ***

(Intercept)
   2.003559
```

Next, our DIY likelihood solution for the regular Poisson counts.

```
# Definition of negative log-likelihood (NLL) for Poisson GLM
NLL1 <- function(param, y, x) {
  alpha <- param[1]
  beta <- param[2]
  lambda <- exp(alpha + beta * x)
  LL <- dpois(y, lambda, log = TRUE)      # LL contribution for each datum
  return(-sum(LL))                        # NLL for all data
}
```

```
# Minimize NLL
inits <- c(alpha = 0, beta = 0)          # Need to provide initial values
sol1 <- optim(inits, NLL1, y = C1, x = selev1, hessian = TRUE, method = 'BFGS')
tmp1 <- get_MLE(sol1, 4)
```

```
           MLE       ASE   LCL.95    UCL.95
alpha    0.6949   0.03668    0.623    0.7668
beta    -1.9613   0.06700   -2.093   -1.8300
```

Next, the same in JAGS ...

```
# Bundle data
str(bdata <- list(C1 = C1, nsites1 = nsites1, selev1 = selev1))
```

```
List of 3
 $ C1     : int [1:500] 8 7 6 8 12 18 5 8 7 8 ...
 $ nsites1: num 500
 $ selev1 : num [1:500] -0.8 -0.797 -0.795 -0.788 -0.782 ...
```

```
# Write JAGS model file
cat(file = "model1.txt", "
model {
# Priors and linear models
alpha ~ dunif(-10, 10)                    # Abundance intercept on log scale
mean.lam <- exp(alpha)                    # Abundance intercept on natural scale
beta ~ dnorm(0, 0.0001)                   # Slope on elevation

# Likelihood for data set 1
for (i in 1:nsites1){
  C1[i] ~ dpois(lambda1[i])
  log(lambda1[i]) <- alpha + beta * selev1[i]
}
}
")

# Initial values
inits <- function(){list(alpha = runif(1), beta = rnorm(1))}

# Parameters monitored
params <- c("mean.lam", "alpha", "beta")

# MCMC settings
ni <- 6000; nb <- 2000; nc <- 4; nt <- 4; na <- 1000

# Call JAGS from R (ART <1 min), check convergence and summarize posteriors
out1 <- jags(bdata, inits, params, "model1.txt", n.iter = ni, n.burnin = nb, n.chains = nc,
  n.thin = nt, n.adapt = na, parallel = TRUE)
jagsUI::traceplot(out1)                          # Not shown
print(out1, 4)
```

```
             mean      sd     2.5%      50%    97.5% overlap0 f    Rhat  n.eff
mean.lam    2.0034  0.0725   1.8620   2.0034   2.1476    FALSE 1  1.0006   3371
alpha       0.6942  0.0362   0.6216   0.6949   0.7643    FALSE 1  1.0006   3541
beta       -1.9599  0.0669  -2.0910  -1.9589  -1.8311    FALSE 1  0.9999   4000
```



```
# Compare truth with likelihood and Bayesian solutions
comp <- cbind(truth = truth, tmp1[,1:2], out1$summary[2:3, 1:2])
colnames(comp)[3:5] <- c("SE(MLE)", "Post.mean", "Post.sd")
print(comp, 3)
```

```
          truth      MLE   SE(MLE)   Post.mean   Post.sd
log.lam   0.693    0.695    0.0367       0.694    0.0362
beta     -2.000   -1.961    0.0670      -1.960    0.0669
```

We find practically matching estimates, as we would expect for this simple model, our use of vague priors, and the large sample size.

20.3.2 Fitting the zero-truncated Poisson GLM in R and JAGS to data set 2

Next, we want to fit the same model to the zero-truncated data set (number 2). This model could be fit using R package VGAM (Yee et al., 2008), but here we just define the zero-truncated Poisson likelihood function ourselves and maximize it. This likelihood is formed as a re-normalized Poisson likelihood without the zero class. That is, we must divide the standard Poisson PMF by 1 minus the probability of a zero count under that same PMF. The latter bit is the total probability of the zero-truncated random variable under the Poisson with same parameters.

```
# Definition of negative log-likelihood (NLL) for the ztPoisson GLM
NLL2 <- function(param, y, x) {
  alpha <- param[1]
  beta <- param[2]
  lambda <- exp(alpha + beta * x)
  L <- dpois(y, lambda)/(1-ppois(0, lambda))     # L. contribution for each datum
  return(-sum(log(L)))                           # NLL for all data
}

# Minimize NLL
inits <- c(alpha = 0, beta = 0)            # Need to provide initial values
sol2 <- optim(inits, NLL2, y = ztC2, x = selev2[C2>0], hessian = TRUE, method = 'BFGS')
tmp2 <- get_MLE(sol2, 4)
```

```
           MLE       ASE    LCL.95    UCL.95
alpha    0.6461   0.03852    0.5706    0.7216
beta    -2.0351   0.07089   -2.1740   -1.8961
```

As an aside, we quickly compare with the solution when we ignore the truncation and erroneously fit the regular Poisson model. We see that we greatly overestimate abundance when we ignore the zero truncation by fitting a standard Poisson distribution to the zero-truncated data.

```
# Minimize 'wrong' Poisson NLL which ignores the zero truncation
sol2a <- optim(inits, NLL1, y = ztC2, x = selev2[C2>0], hessian = TRUE, method = 'BFGS')
get_MLE(sol2a, 4)
```

```
           MLE       ASE    LCL.95    UCL.95
alpha    0.9276   0.02755    0.8736    0.9816
beta    -1.5275   0.05402   -1.6334   -1.4216
```

```
# Compare intercepts of right and wrong Poisson NLL on the natural scale
exp(sol2$par[1]) ; exp(sol2a$par[1])

   alpha
1.908095
   alpha
2.528512
```

Next, we fit the zero-truncated Poisson model in JAGS, where it's particularly easy to fit this model, using the `T()` syntax. We are reminded that by zero truncating we lost the data from almost 300 sites in our simulated data set. We index sites by *j*, to emphasize that we assume a different set of sites in each of the three data sets.

```
# Bundle data
str(bdata <- list(C2 = ztC2, nsites2 = length(ztC2), selev2 = selev2[C2 > 0]))

List of 3
 $ C2     :  int [1:728]  11  13  5  9  8  7  14  6  7  6  ...
 $ nsites2:  int 728
 $ selev2 :  num [1:728]  -0.799  -0.798  -0.798  -0.797  -0.797  ...

# Write JAGS model file
cat(file = "model2.txt", "
model {
# Priors and linear models
alpha ~ dunif(-10, 10)                    # Abundance intercept on log scale
mean.lam <- exp(alpha)                    # Abundance intercept on natural scale
beta ~ dnorm(0, 0.0001)                   # Slope on elevation

# Zero-truncated Poisson likelihood for data set 2
for (j in 1:nsites2){
  C2[j] ~ dpois(lambda1[j])T(1,)          # truncation is accommodated easily
  log(lambda1[j]) <- alpha + beta * selev2[j]
}
}
")

# Initial values
inits <- function(){list(alpha = runif(1), beta = rnorm(1))}

# Parameters monitored
params <- c("mean.lam", "alpha", "beta")

# MCMC settings
ni <- 6000; nb <- 2000; nc <- 4; nt <- 4; na <- 1000

# Call JAGS from R (ART <1 min), check convergence and summarize posteriors
out2 <- jags(bdata, inits, params, "model2.txt", n.iter = ni, n.burnin = nb,
  n.chains = nc, n.thin = nt, n.adapt = na, parallel = TRUE)
jagsUI::traceplot(out2)                   # Not shown
print(out2, 2)

           mean    sd    2.5%     50%   97.5%  overlap0  f  Rhat  n.eff
mean.lam   1.91  0.07    1.76    1.90    2.05     FALSE  1     1   1219
alpha      0.64  0.04    0.57    0.64    0.72     FALSE  1     1   1226
beta      -2.04  0.07   -2.18   -2.04   -1.90     FALSE  1     1   1887
```


We compare the likelihood and the Bayesian estimates.

```
# Compare truth with likelihood and Bayesian solutions
comp <- cbind(truth = truth, tmp2[,1:2], out2$summary[2:3, 1:2])
colnames(comp)[3:5] <- c("SE(MLE)", "Post.mean", "Post.sd")
print(comp, 3)

          truth      MLE   SE(MLE)   Post.mean   Post.sd
log.lam   0.693    0.646    0.0385       0.644    0.0391
beta     -2.000   -2.035    0.0709      -2.039    0.0719
```

That looks good!

20.3.3 Fitting the cloglog Bernoulli GLM in R and JAGS to data set 3

Finally, we fit the Bernoulli model with cloglog link to the detection/nondetection data. This means that the parameters of the model describe the relationship between an underlying abundance process and the elevation covariate.

```
# Data set 3: Bernoulli GLM with cloglog link for detection/nondetection
summary(fm3 <- glm(y ~ selev3, family = binomial(link = "cloglog")))
exp(coef(fm3)[1])   # Estimate of lambda on natural scale from binary data

Coefficients:
            Estimate   Std. Error   z value   Pr(>|z|)
(Intercept)  0.71081      0.04169     17.05     <2e-16 ***
selev3      -1.98144      0.09499    -20.86     <2e-16 ***
---

(Intercept)
   2.035632
```

And the DIY likelihood solution.

```
# Definition of NLL for Bernoulli GLM with cloglog link
NLL3 <- function(param, y, x) {
  alpha <- param[1]
  beta <- param[2]
  lambda <- exp(alpha + beta * x)
  psi <- 1-exp(-lambda)
  LL <- dbinom(y, 1, psi, log = TRUE)             # L. contribution for each datum
  return(-sum(LL))                                # NLL for all data
}

# Minimize NLL
inits <- c(alpha = 0, beta = 0)
sol3 <- optim(inits, NLL3, y = y, x = selev3, hessian = TRUE, method = 'BFGS')
tmp3 <- get_MLE(sol3, 4)

           MLE      ASE    LCL.95    UCL.95
alpha   0.7108  0.04214    0.6282    0.7934
beta   -1.9815  0.09682   -2.1712   -1.7917
```

And finally JAGS. In the model, we now index sites by *k*.

```
# Bundle data
str(bdata <- list(y = y, nsites3 = nsites3, selev3 = selev3))

List of 3
 $ y      : num [1:2000] 1 1 1 1 1 1 1 1 1 1 ...
 $ nsites3: num 2000
 $ selev3 : num [1:2000] -0.798 -0.794 -0.794 -0.794 -0.792 ...

# Write JAGS model file
cat(file = "model3.txt", "
model {
# Priors and linear models
alpha ~ dunif(-10, 10)                    # Abundance intercept on log scale
mean.lam <- exp(alpha)                    # Abundance intercept on natural scale
beta ~ dnorm(0, 0.0001)                   # Slope on elevation

# Likelihood for data set 3
for (k in 1:nsites3){
  y[k] ~ dbern(psi[k])
  cloglog(psi[k]) <- alpha + beta * selev3[k]

  # Alternative implementation of same model for data set 3
  # y[k] ~ dbern(psi[k])
  # psi[k] <- 1 - exp(-lambda3[k])
  # log(lambda3[k]) <- alpha + beta * selev3[k]
}
}
")

# Initial values
inits <- function(){list(alpha = runif(1), beta = rnorm(1))}

# Parameters monitored
params <- c("mean.lam", "alpha", "beta")

# MCMC settings
ni <- 6000; nb <- 2000; nc <- 4; nt <- 4; na <- 1000

# Call JAGS from R (ART 1.2 min), check convergence and summarize posteriors
out3 <- jags(bdata, inits, params, "model3.txt", n.iter = ni, n.burnin = nb,
  n.chains = nc, n.thin = nt, n.adapt = na, parallel = TRUE)
jagsUI::traceplot(out3)                   # Not shown
print(out3, 3)

            mean      sd     2.5%      50%    97.5%  overlap0    f   Rhat  n.eff
mean.lam   2.040   0.084    1.885    2.038    2.211     FALSE   1  1.002   1439
alpha      0.712   0.041    0.634    0.712    0.794     FALSE   1  1.002   1413
beta      -1.986   0.095   -2.177   -1.985   -1.804     FALSE   1  1.002   1240

# Compare truth with likelihood and Bayesian solutions
comp <- cbind(truth = truth, tmp3[,1:2], out3$summary[2:3, 1:2])
colnames(comp)[3:5] <- c("SE(MLE)", "Post.mean", "Post.sd")
print(comp, 3)

          truth      MLE   SE(MLE)   Post.mean   Post.sd
log.lam   0.693    0.711    0.0421       0.712    0.0411
beta     -2.000   -1.981    0.0968      -1.986    0.0948
```

We find a nice numerical agreement.

20.4 Fitting the integrated model to all three data sets simultaneously

Now that we understand the component models for each data set and know how to code them up, we are ready to fit the IM. We fit the full integrated SDM which assumes a shared abundance process, but accommodates the three different types of observation processes in the three simulated data sets. There is no canned R function for us to fit this IM, so we work directly with our own negative log-likelihood function that we then maximize for the three data sets simultaneously. After that, we fit the same model with JAGS, NIMBLE, Stan, and TMB.

20.4.1 Fitting the IM with a DIY likelihood function

Under an assumption of statistical independence the joint likelihood for the three submodels is simply the product of the three single-data likelihoods:

$$L_{joint} = L_1 L_2 L_3$$

Of course, we usually work with the negative of the log of the likelihood and for this, we have a sum rather than a product. Hence, the objective function to minimize for our IM is this:

$$NLL_{joint} = -(LL_1 + LL_2 + LL_3) = -LL_1 - LL_2 - LL_3$$

Let's try this out.

```
# Definition of the joint NLL for the integrated model
NLLjoint <- function(param, y1, x1, y2, x2, y3, x3) {
  # Definition of elements in param vector (shared between data sets)
    alpha <- param[1]                                   # log-linear intercept
    beta <- param[2]                                    # log-linear slope
  # Likelihood for the Poisson GLM for data set 1 (y1, x1)
    lambda1 <- exp(alpha + beta * x1)
    L1 <- dpois(y1, lambda1)
  # Likelihood for the ztPoisson GLM for data set 2 (y2, x2)
    lambda2 <- exp(alpha + beta * x2)
    L2 <- dpois(y2, lambda2)/(1-ppois(0, lambda2))
  # Likelihood for the cloglog Bernoulli GLM for data set 3 (y3, x3)
    lambda3 <- exp(alpha + beta * x3)
    psi <- 1-exp(-lambda3)
    L3 <- dbinom(y3, 1, psi)
  # Joint log-likelihood and joint NLL: here you can see that sum!
    JointLL <- sum(log(L1)) + sum(log(L2)) + sum(log(L3))    # Joint LL
    return(-JointLL)                                    # Return joint NLL
}

# Minimize NLLjoint
inits <- c(alpha = 0, beta = 0)
solJoint <- optim(inits, NLLjoint, y1 = C1, x1 = selev1, y2 = ztC2, x2 = selev2[C2>0],
  y3 = y, x3 = selev3, hessian = TRUE, method = 'BFGS')

# Get MLE and asymptotic SE and print and save
(tmp4 <- get_MLE(solJoint, 4))
diy_est <- tmp4[,1]

              MLE         ASE       LCL.95       UCL.95
alpha   0.6860979  0.01851556   0.6498074    0.7223884
beta   -1.9703732  0.03522001  -2.0394044   -1.9013420
```

Nice... our first IM "by hand"!

20.4.2 Fitting the IM with JAGS

For the IM in JAGS, we simply stack the three GLMs inside of the same BUGS model statement and choose the same names for the parameters `alpha` and `beta` in all of them. This is what defines the joint likelihood as a product of the single-data likelihoods. Hence, even though you don't see it so clearly, the likelihood of this model is $L_{joint} = L_1 L_2 L_3$, where L_1, L_2, and L_3 are, respectively, the likelihoods of the Poisson GLM for the count data set 1, of the zero-truncated Poisson GLM for data set 2, and of the Bernoulli GLM for the detection/nondetection data set 3. Note again the different indices used for sites in the three data sets to emphasize that we assume they do not overlap.

```
# Bundle data
str(dataList <- list(C1 = C1, C2 = ztC2, y = y, nsites1 = nsites1, nsites2 = length(ztC2),
  nsites3 = nsites3, selev1 = selev1, selev2 = selev2[C2>0], selev3 = selev3))

List of 9
 $ C1     :  int  [1:500]  8  7  6  8  12  18  5  8  7  8  ...
 $ C2     :  int  [1:728]  11  13  5  9  8  7  1  4  6  7  6...
 $ y      :  num  [1:2000]  1  1  1  1  1  1  1  1  1  1  ...
 $ nsites1:  num  500
 $ nsites2:  int  728
 $ nsites3:  num  2000
 $ selev1 :  num  [1:500]  -0.8  -0.797  -0.795  -0.788  -0.782  ...
 $ selev2 :  num  [1:728]  -0.799  -0.798  -0.798  -0.797  -0.797  ...
 $ selev3 :  num  [1:2000]  -0.798  -0.794  -0.794  -0.794  -0.792  ...

# Write JAGS model file
cat(file = "model4.txt", "
model {
# Priors and linear models: shared for models of all three data sets
alpha ~ dunif(-10, 10)                  # Abundance intercept on log scale
mean.lam <- exp(alpha)                  # Abundance intercept on natural scale
beta ~ dnorm(0, 0.0001)                 # Slope on elevation

# Joint likelihood: Note identical alpha and beta for all data sets
# Likelihood portion for data set 1: regular counts
for (i in 1:nsites1){
  C1[i] ~ dpois(lambda1[i])
  log(lambda1[i]) <- alpha + beta * selev1[i]
}
# Likelihood portion for data set 2: zero-truncated counts
for (j in 1:nsites2){
  C2[j] ~ dpois(lambda2[j])T(1,)
  log(lambda2[j]) <- alpha + beta * selev2[j]
}
# Likelihood portion for data set 3: detection/nondetection
for (k in 1:nsites3){
  y[k] ~ dbern(psi[k])
  cloglog(psi[k]) <- alpha + beta * selev3[k]
}
}
")
```



```
# Initial values
inits <- function(){list(alpha = runif(1), beta = rnorm(1))}

# Parameters monitored
params <- c("mean.lam", "alpha", "beta")

# MCMC settings
na <- 1000; ni <- 6000; nb <- 2000; nc <- 4; nt <- 4

# Call JAGS from R (ART 170 sec), check convergence and summarize posteriors
out4 <- jags(dataList, inits, params, "model4.txt", n.iter = ni, n.burnin = nb,
  n.chains = nc, n.thin = nt, n.adapt = na, parallel = TRUE)
jagsUI::traceplot(out4)
print(out4, 2)
jags_est <- out4$summary[2:3,1]

          mean   sd  2.5%   50% 97.5% overlap0 f Rhat n.eff
mean.lam  1.99 0.04  1.91  1.99  2.06    FALSE 1    1  4000
alpha     0.69 0.02  0.65  0.69  0.72    FALSE 1    1  4000
beta     -1.97 0.04 -2.04 -1.97 -1.90    FALSE 1    1  4000
```

We compare the likelihood and the Bayesian solutions with truth.

```
# Compare truth with likelihood and Bayesian solutions
comp <- cbind(truth = truth, tmp4[,1:2], out4$summary[2:3, 1:2])
colnames(comp)[3:5] <- c("SE(MLE)", "Post.mean", "Post.sd")
print(comp, 3)

          truth      MLE   SE(MLE)   Post.mean   Post.sd
log.lam   0.693    0.686    0.0185       0.686    0.0184
beta     -2.000   -1.970    0.0352      -1.971    0.0353
```

As almost always, we get numerically extremely similar estimates from maximum likelihood and Bayesian posterior inference for a model with vague priors and provided that the sample size is large with respect to the complexity of the model.

20.4.3 Fitting the IM with NIMBLE

NIMBLE code for the model is available on the book website.

20.4.4 Fitting the IM with Stan

Also for Stan, we can use the same bundled data. Here's the model code, which looks very similar to our JAGS model.

```
cat(file = "model4.stan",
"data {
  int nsites1;
  int nsites2;
  int nsites3;
  array[nsites1] int C1;
  array[nsites2] int C2;
  array[nsites3] int y;
  vector[nsites1] selev1;
  vector[nsites2] selev2;
  vector[nsites3] selev3;
}
parameters {
  real alpha;
  real beta;
}
model {
  vector[nsites1] lambda1;
  vector[nsites2] lambda2;
  vector[nsites3] psi;

  // Priors
  alpha ~ uniform(-10, 10);
  beta ~ normal(0, 100);

  // Likelihood
  for (i in 1:nsites1){
    lambda1[i] = exp(alpha + beta * selev1[i]);
    C1[i] ~ poisson(lambda1[i]);
  }
  for (j in 1:nsites2){
    lambda2[j] = exp(alpha + beta * selev2[j]);
    C2[j] ~ poisson(lambda2[j]) T[1, ];
  }
  for (k in 1:nsites3){
    psi[k] = inv_cloglog(alpha + beta * selev3[k]);
    y[k] ~ bernoulli(psi[k]);
  }
}
generated quantities {
  real mean_lam = exp(alpha);
}
")

# Parameters monitored
params <- c("mean_lam", "alpha", "beta")

# HMC settings
ni <- 2000  ; nb <- 1000  ; nc <- 4  ; nt <- 1

# Call STAN (ART 90/45 sec), assess convergence and print results table
system.time(
out4.stan <- stan(file = "model4.stan", data = dataList,
                  warmup = nb, iter = ni, chains = nc, thin = nt) )
rstan::traceplot(out4.stan)                    # not shown
print(out4.stan, dig = 3)                      # not shown
stan_est <- summary(out4.stan)$summary[1:2,1]
```


20.4.5 Fitting the IM with TMB

Again we can use the same data bundle.

```
cat(file = "model4.cpp",
"#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  //Describe input data
  DATA_INTEGER(nsites1);
  DATA_INTEGER(nsites2);
  DATA_INTEGER(nsites3);
  DATA_VECTOR(C1);
  DATA_VECTOR(C2);
  DATA_VECTOR(y);
  DATA_VECTOR(selev1);
  DATA_VECTOR(selev2);
  DATA_VECTOR(selev3);

  //Describe parameters
  PARAMETER(alpha);
  PARAMETER(beta);

  Type LL = 0.0;                                  //Initialize log-likelihood at 0

  vector<Type> lambda1(nsites1);
  vector<Type> lambda2(nsites2);
  vector<Type> psi(nsites3);

  for (int i=0; i<nsites1; i++){
    lambda1(i) = exp(alpha + beta * selev1(i));
    LL += dpois(C1(i), lambda1(i), true);
  }
  for (int j = 0; j<nsites2; j++){
    lambda2(j) = exp(alpha + beta * selev2(j));
    // Truncated Poisson
    LL += log(dpois(C2(j), lambda2(j))/(1 - ppois(Type(0), lambda2(j))));
  }
  for (int k=0; k<nsites3; k++){
    psi(k) = 1 - exp(-exp(alpha + beta * selev3(k))); //inverse cloglog
    LL += dbinom(y(k), Type(1), psi(k), true);
  }

  Type mean_lam = exp(alpha);
  ADREPORT(mean_lam);

  return -LL;
}
")
```

```
# Compile and load TMB function
compile("model4.cpp")
dyn.load(dynlib("model4"))

# Provide dimensions and starting values for parameters
params <- list(alpha = 0, beta = 0)

# Create TMB object
out4.tmb <- MakeADFun(data = dataList, parameters = params,
                      DLL = "model4", silent = TRUE)

# Optimize TMB object, print and save results
opt <- optim(out4.tmb$par, fn = out4.tmb$fn, gr = out4.tmb$gr, method = "BFGS")
(tsum <- tmb_summary(out4.tmb))
tmb_est <- tsum[1:2,1]
```

20.4.6 Comparison of the parameter estimates for the IM

We compare the parameter estimates for the IMs among the engines. Remember that we have fewer engines now, since there is no canned function in R to fit our IM.

```
# Compare point estimates from the five engines
comp <- cbind(truth = truth, DIY = diy_est, JAGS = jags_est, NIMBLE = nimble_est,
  Stan = stan_est, TMB = tmb_est)
print(comp, 4)
```

```
          truth      DIY     JAGS   NIMBLE     Stan      TMB
log.lam  0.6931   0.6861   0.6855    0.686   0.6856   0.6861
beta    -2.0000  -1.9704  -1.9710   -1.970  -1.9715  -1.9704
```

We see what we see so often: numerically, these estimates are practically indistinguishable.

20.5 What do we gain by analyzing the joint likelihood in our analysis?

We compare all the MLEs for models 1–4 to see how point estimates and asymptotic standard errors change for different data types and for the single-data models and the joint-likelihood model. Since we have already seen the similarity between the likelihood and the Bayesian inferences, we're not going to bother doing this for both, but restrict this comparison to our likelihood inferences.

```
# Compare truth with MLEs only from all 4 models (stacked sideways)
print(cbind(truth = truth, "MLE(Poisson)" = tmp1[,1], "MLE(ZTPois)" = tmp2[,1],
  "MLE(cloglogBern)" = tmp3[,1], "MLE(integrated)" = tmp4[,1]), 3)

# Compare ASEs from all 4 models (stacked sideways)
print(cbind("ASE(Poisson)" = tmp1[,2], "ASE(ZTPois)" = tmp2[,2], "ASE(cloglogBern)" = tmp3[,2],
  "ASE(integrated)" = tmp4[,2]), 3)
```



```
## Point estimates
          truth  MLE(Poisson)  MLE(ZTPois)  MLE(cloglogBern)  MLE(integrated)
log.lam   0.693         0.695        0.646             0.711            0.686
beta     -2.000        -1.961       -2.035            -1.981           -1.970

## Standard errors
       ASE(Poisson)  ASE(ZTPois)  ASE(cloglogBern)  ASE(integrated)
alpha        0.0367       0.0385            0.0421           0.0185
beta         0.0670       0.0709            0.0968           0.0352
```

We don't see much difference among the point estimates. However, we see that the model for the regular counts has the greatest precision among the single-data models, followed by that for the zero-truncated counts, while that for the detection/nondetection data comes last. This makes intuitive sense, since we reduce the information by going from regular counts to tossing out the zeroes only and finally quantizing the data. In our simulation, this information loss is not compensated for by our simulated increase in sample size.

The IM has the greatest precision of all, since it uses the information from all three data sets in combination. We also make a plot to compare the point estimates and CRIs of models 1–4 (Fig. 20.4).

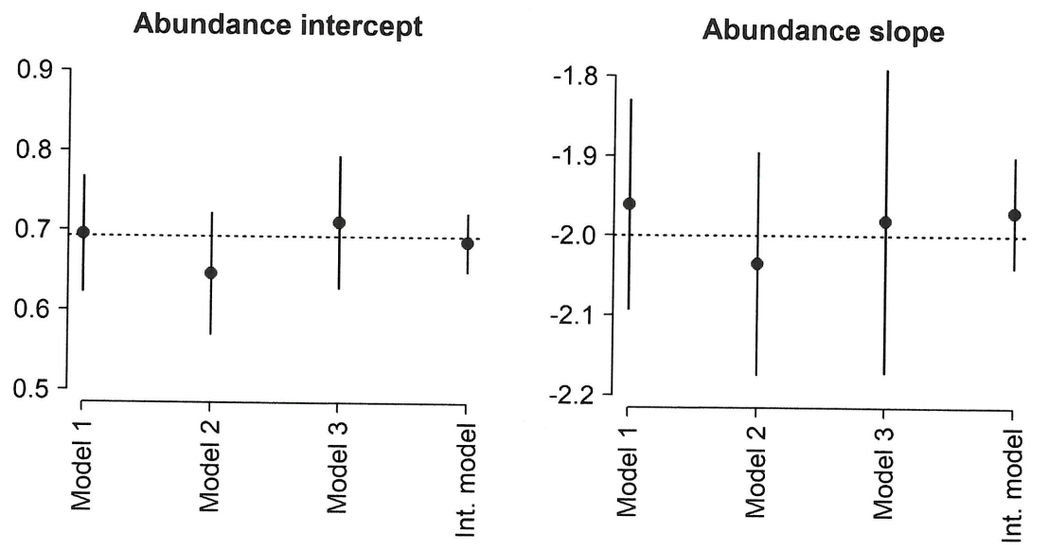

FIGURE 20.4

Point estimates (MLEs) and 95% CIs for the intercept and the slope parameter in the abundance model in all four models (dotted horizontal lines show truth). For both parameters, the precision of the estimates is greatest when we use all the information available, which is what we achieve by the integrated model.

```
# Compare MLEs and ASEs from all 4 models (Fig. 20.4)
par(mfrow = c(1, 2), mar = c(12, 6, 5, 3), cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.8)
# Plot for abundance intercept (on log scale)
all.mles <- c(tmp1[1,1], tmp2[1,1], tmp3[1,1], tmp4[1,1])
all.lower.CL <- c(tmp1[1,3], tmp2[1,3], tmp3[1,3], tmp4[1,3])
all.upper.CL <- c(tmp1[1,4], tmp2[1,4], tmp3[1,4], tmp4[1,4])
plot(1:4, all.mles, pch = 16, xlab = '', ylab = '', axes = FALSE, frame = FALSE,
  main = 'Abundance intercept', cex = 2, ylim = c(0.5, 0.9))
axis(1, at = 1:4, c('Model 1', 'Model 2', 'Model 3', 'Integrated model'), las = 2)
segments(1:4, all.lower.CL, 1:4, all.upper.CL, lwd = 1.5)
axis(2, las = 1)
abline(h = log(2), lwd = 1, lty = 3)

# Plot for abundance slope (on log scale)
all.mles <- c(tmp1[2,1], tmp2[2,1], tmp3[2,1], tmp4[2,1])
all.lower.CL <- c(tmp1[2,3], tmp2[2,3], tmp3[2,3], tmp4[2,3])
all.upper.CL <- c(tmp1[2,4], tmp2[2,4], tmp3[2,4], tmp4[2,4])
plot(1:4, all.mles, pch = 16, xlab = '', ylab = '', axes = FALSE, frame = FALSE,
  main = 'Abundance slope', cex = 2, ylim = c(-2.2, -1.8))
axis(1, at = 1:4, c('Model 1', 'Model 2', 'Model 3', 'Integrated model'), las = 2)
segments(1:4, all.lower.CL, 1:4, all.upper.CL, lwd = 1.5)
axis(2, las = 1)
abline(h = -2, lwd = 1, lty = 3)
```

In Fig. 20.4, we see that the joint likelihood model is best in terms of the precision for both parameters in our model. We note that all estimators (i.e., all models) should produce unbiased estimates. Hence, it would be wrong to say that the differences between point estimates and the true values represented by the dotted lines represent bias. To gauge the unbiasedness of an estimator, we would have to repeat the whole data simulation/data analysis cycle a large number of times (e.g., 100 or 1000 times) and then look at the distribution of these estimates. Such a simulation could easily be done, especially with MLEs, since that method is much faster than MCMC.

The IM in this chapter does not illustrate another common advantage of this model class: namely that additional parameters may become estimable. For instance, in an IPM with a time series of counts that is combined with capture–recapture data we can estimate productivity, which is not an identifiable parameter in separate analyses for either data set alone (Besbeas et al., 2002; Schaub & Kéry, 2022).

20.6 Summary and outlook

Now you understand the principle of data integration by working with a joint likelihood: we first identify some underlying process that is shared among the different data sets and can be expressed by one or several parameters, and then describe the differences between the different data sets that typically lie in the observation process. In our case, even though at the start we had abundance data (i.e., counts) and detection/nondetection data, we assumed an underlying abundance process that is shared for all three data sets. Then, we chose a likelihood that represented the different observation processes that underlie the three different data sets: a regular Poisson PMF for the proper counts, a zero-truncated Poisson PMF for the data produced by the "lazy birders," and a Bernoulli PMF with

cloglog link for the detection/nondetection data. Assuming statistical independence, the joint likelihood is the product of the likelihoods of each individual data set. Maximizing it results in parameter estimates that are most likely with respect to all data sets in the model.

Seeing that the joint likelihood under independence is a product of the single-data likelihoods is easiest when working with DIY maximum likelihood, because here the joint likelihood is very clearly apparent. However, in practice for most non-statisticians it is much easier to specify that same joint likelihood for an IM in the BUGS language or even with Stan or TMB, where we simply describe the individual-data likelihoods within the same model statement.

Opportunities for adopting such IMs are superabundant in ecology and wildlife management. Clearly, we expect to see more and more models that apply the principles of integrated modeling. The power and flexibility of the BUGS language makes it almost trivially easy to define the joint likelihood for such an IM. IMs are almost always custom models to some degree and it is here where the new-found modeling freedom that especially BUGS software gives you will be particularly exciting for you.