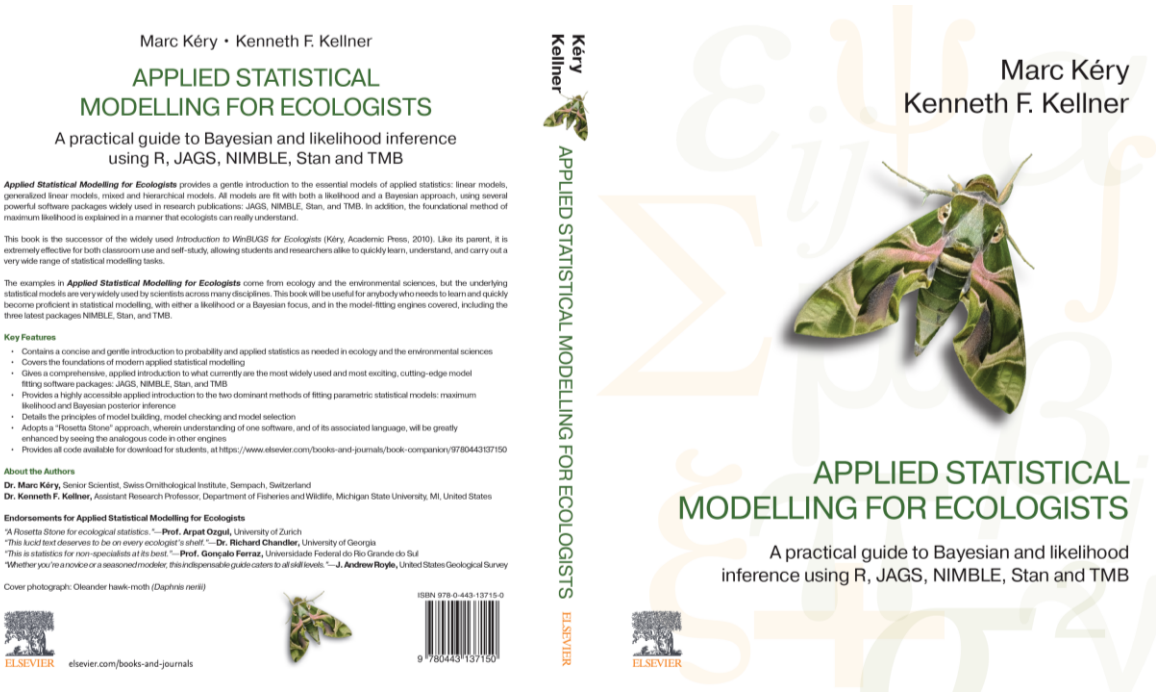
Bonus Chapter 19B of Applied Statistical Modeling for Ecologists (ASM)

Your gentle introduction and quick-start guide to:

- ***the Bayesian approach to statistical modeling***
- ***the essentials of applied statistical modeling: linear, generalized linear and mixed models***
- ***the latest, cutting-edge statistical software JAGS, NIMBLE, Stan and TMB***
- ***maximum likelihood estimation for all the above models in practice***



Marc Kéry & Ken Kellner,
Swiss Ornithological Institute & MSU

6 July 2024

This is a bonus chapter of the *Applied Statistical Modeling* (ASM) book by Kéry & Kellner (Elsevier, 2024). It does not appear in the printed book, but is available on the book website at https://www.elsevier.com/books-and-journals/book-companion/9780443137150. Parts of the content draw from chapter 21 in the little blue bugs book (Kéry, 2010).

Table of Contents

19B Binomial *N*-mixture models

Key topics: abundance, delta method, density, detection/nondetection data, detection probability, discrete random effects, false negative observation error, integrated likelihood, marginal likelihood, measurement error, observation error, observation process, package `unmarked`, package `ubms`, parametric bootstrap, repeated-measures design, species distribution model (SDM)

19B.1 Introduction

Ecology has been defined as the study of distribution and abundance (Andrewartha & Birch 1954; Krebs 2009). However, in nature neither of them can usually be observed without error, and statistical methods need to be applied to infer the true states of distribution and abundance from error-prone observations. In Chapter 19 we met a repeated-measures protocol where detection or nondetection of a species was assessed across M sites and T temporal replicates. This design enabled the application of an occupancy model to estimate the true species distribution, as represented by occupancy probability, free of the distorting effects of imperfect detection. Temporal replicate observations in a closed system allowed us to resolve the confounding between occupancy and detection. The occupancy model is a hierarchical model that is in a sense similar to a binomial GLMM (Chapter 17) but has binary random effects that do not appear in the linear predictor in an additive fashion. Most of all, in contrast to a GLMM, these random effects have a clear biological meaning: they denote the presence or absence of the species at each site.

This chapter showcases another hierarchical model with a non-standard random effects distribution, namely Poisson. As in the occupancy model, the random effects in this model also have a precise biological meaning, which now is local population size (Royle & Dorazio, 2008). Thus, with this model we can estimate abundance corrected for imperfect detection based on temporally and spatially replicated *counts*, rather than detection/nondetection data as in the occupancy model.

The ecological motivation in this chapter is the modeling of the abundance of a songbird species, the bullfinch (Fig. 19.1), along an elevational gradient in Switzerland. One of our aims is to identify the *optimum elevation* of that species, at which its expected abundance is greatest. This may be a useful single-number characterization of the species distribution in the context of climate change studies, where we might for instance be interested in whether the optimum elevation moves up over time.

We will model replicated counts from the Swiss breeding bird survey MHB (*"Monitoring Häufige Brutvögel"*; Schmid et al. 2004), where a total of 267 1 km^2 sample quadrats are laid out in an approximate grid across the country. Since 1999, each quadrat has been surveyed repeatedly in every breeding season (mid-April to mid-July) along a wiggly transect. This transect aims to cover as much area of the quadrat as possible. It averages about 4–6 km in length and for each quadrat remains identical across all years. During each survey, a territory-mapping protocol (Bibby *et al.* 2000) for measuring abundance is applied, in which an observer records on a map every bird individual detected by sight or ear. Three such *repeated measurements of abundance* are taken in each quadrat per year, except in quadrats at greater than 2000 m elevation, where only two surveys are conducted per season. Within a given quadrat, subsequent surveys within the same breeding season are typically conducted about two weeks apart. This short duration justifies the closure assumption for many of the surveyed species, including the bullfinch. For

more information about territory mapping and the Swiss MHB, see Kéry *et al.* (2005) and Sections 6.9 and 7.9 in Kéry & Royle (2016).

The typical question in biodiversity monitoring programs is always: *"Are things getting better or worse?"*, or, in other words, whether there a trend over time in a biological quantity such as abundance or the number of occupied sites. The usual type of analysis to answer this question for counts is typically a Poisson GLM or GLMM, and examples include Link & Sauer (2002), Ver Hoef & Jansen (2007), Barker et al. (2016), Strebel *et al.* (2020).

Fig. 19B–1: A glorious male bullfinch (*Pyrrhula pyrrhula*, photo by Markus Varesvuo).

When repeated counts are available, a common practice is to only retain and analyze the maximum count, although this in fact throws out valuable information. In this chapter, we showcase a model that makes full use of replicated counts: the binomial *N*-mixture model of Royle (2004a). This model yields estimates of true abundance that are corrected for imperfect detection; see also Wyatt (2002), Dodd & Dorazio (2004), Royle (2004b), Kéry *et al.* (2005), Royle *et al.* (2005), Dorazio (2007), Kéry (2008), Wenger & Freeman (2008), Knaus *et al.* (2018), Madsen & Royle (2023). For simplicity, we will simulate data from a single breeding season and assume population closure. Variants of the *N*-mixture model exist for multi-year data, and they enable direct estimation of population trends in abundance (Royle & Dorazio, 2008, p. 4–7; Kéry *et al.* 2009; Kéry & Royle 2010; Dail & Madsen, 2011). For recent monographs on various static and dynamic types of *N*-mixture models, see chapters 6–8 in Kéry & Royle (2016) and chapters 1 and 2 in Kéry & Royle (2021).

We use the term 'site' to denote the spatial sampling units, which in our example are 1 km^2 quadrats. Other examples of sites include point counts or transects, or more naturally defined spatial units such as ponds, nature reserves, small woodlots, or backyards. The *N*-mixture model may be applicable to the resulting data which are replicated in space (i.e., over sites) and in time

(i.e., over repeated measurements), provided sites are well-defined and population size at each site is constant over the study duration for which closure is assumed.

So, let's now have a look at the static *N*-mixture model for repeated abundance measurements (also called surveys, or occasions). We'll show the model for a set of closed populations (i.e., at each site) in a single year, matching the structure of a single year of MHB data. We assume that bird count $y_{i,t}$ at site i made during survey t comes from a two-stage stochastic process. The first stochastic process is the biological process that distributes the animals in space, i.e., among sites. This process generates the site-specific abundance that we would like to model directly, but cannot, because we will arguably never see all individual animals or plants in a field survey. The typical statistical model for such abundance data is the Poisson distribution, which is governed by the intensity, or density, parameter λ, which in turn is typically expressed conditional on habitat covariates. The result of this first stochastic process is the local, site-specific abundance N_i.

The second stochastic process is an observation process, which is expressed conditional on, or given, the true state N_i of a site. The observation process together with N_i determines the data actually observed, i.e., the counts $y_{i,t}$. In the absence of any double counts, a natural model for the observation process in the presence of imperfect detection is the binomial distribution. Thus, we assume that given that there are N_i bullfinches present at site i and that each has a probability of $p_{i,t}$ to be detected at site i during replicate survey t, the number of finches actually observed during a survey is binomially distributed. Two important consequences are that (1) we cannot observe *more* than N_i finches and typically observe fewer than N_i and (2) counts $y_{i,t}$ will vary from survey to survey even under identical conditions (Kéry & Schmidt, 2008). Note that we won't be able to estimate a separate value of detection p at every site and survey. Hence, we will have to constrain $p_{i,t}$ by making it constant over the spatial or temporal dimensions or by adopting linear models with covariates; see below.

Four important assumptions of the *N*-mixture model are:

- population closure,
- independent and identical detection probability for all individuals at site i and during survey t,
- the absence of any double counts and other sources of false positive errors, and
- that the spatial variation in abundance N_i is adequately described by the assumed statistical distribution, such as a Poisson, zero-inflated Poisson or negative binomial (Chapter 6 in Kéry & Royle, 2016).

The effects of violations of these assumptions and some potential challenges encountered when fitting the model have been studied by various authors, including Joseph *et al.* (2009), Couturier *et al.* (2013), Dennis *et al.* (2015), Duarte *et al.* (2016), Kéry & Royle (2016: Section 6.7), Barker *et al.* (2018), Link *et al.* (2018), Knape *et al.* (2018), and Kéry (2018). See also Section 19B.12 for further comments.

In summary, the simplest binomial *N*-mixture model to estimate abundance from temporally and spatially replicated counts can be written succinctly in just two lines:

$$N_i \sim Poisson(\lambda) \qquad \text{Biological process yields true state}$$
$$y_{i,t} \sim Binomial(N_i, p) \qquad \text{Observation process yields observations}$$

We find it fascinating to note the similarity of this hierarchical species distribution model model (Royle & Dorazio, 2006, 2008) for abundance and the one for occurrence that we saw in

Chapter 19, i.e., the occupancy model. Recognizing that in the occupancy model the observation process for detection frequency data may also be described by a binomial distribution (rather than by a Bernoulli), the sole thing that changes when we go from the modeling of occurrence to that of abundance is the distribution used to model the biological process: a Poisson instead of a Bernoulli. The binomial mixture model can be described as a binomial mixed-effects model that is in some sense similar to a GLMM but has discrete (Poisson-distributed) random effects rather than the normal random effects that are required for a GLMM. Alternatively, we could view the binomial mixture model as a logistic regression for the observed counts, coupled with a Poisson regression for the imperfectly observed abundances.

As in the occupancy model, covariate effects can be added in the *N*-mixture model by modeling the Poisson parameter λ via a log link function and the binomial success rate p via the logit link. Thus, we can add to the model expressions such as $\log(\lambda_i) = \alpha + \beta x_i$ and $\text{logit}(p_{i,t}) = \alpha + \beta_1 x_i^{(1)} + \beta_2 x_{i,t}^{(2)}$, where x_i and $x_i^{(1)}$ are the values of a site covariate measured at site i, and $x_{i,t}^{(2)}$ contains the values of a survey covariate measured at site i during survey t. It is also possible to model distinct effects of the same covariate in the abundance and in the detection parts of the model (Kéry 2008).

19B.2 Data generation

We will simulate replicate bullfinch counts within the 267 1 km^2 quadrats in the Swiss MHB and focus on the elevational gradient of bullfinch abundance. For fun, we work with the actual average quadrat elevation in the MHB, which is available in the `crossbill` data set in the `unmarked` package. The original data are expressed in meters and rounded to 50 m and we will jitter them some to get nicer pictures.

```
# Load Swiss MHB elevation data for 267 quadrats
library(unmarked)
data(crossbill)
str(crossbill)            # not shown

# Pull out elevation data, jitter some and sort
set.seed(191)
mhbElev <- sort(jitter(crossbill$ele, factor = 2))
summary(mhbElev)

# Histogram of elevation (not shown)
par(mar = c(5,5,5,3), cex.lab = 1.5, cex.axis = 1.5)
hist(mhbElev, breaks = 50, xlim = c(0, 3000),
  main = 'Mean elevation of 267 MHB quadrats in Switzerland')
```

In this chapter we will showcase a special type of predictions: spatial predictions. That is, after fitting a model to the data (which we yet have to simulate), we will predict expected abundance for all of Switzerland, to yield a simple species distribution model, or SDM. For this, we will need the elevation data for all of Switzerland, which are available in the `Switzerland` data set in `unmarked`. We load them and produce a map of Swiss elevation (Fig. 19B.2).

```
# Load the full Swiss landscape data and grab the elevation data
data(Switzerland)
ch <- Switzerland
str(ch)
chElev <- ch$elevation # Swiss elevation data in km units
summary(chElev)

# Make a plot of elevation (Fig. 19B.2)
library(raster)
par(mfrow = c(1, 1), mar = c(3,4,4,8), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = chElev))
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Swiss elevation map (in m)", zlim = c(0, 4500))
```

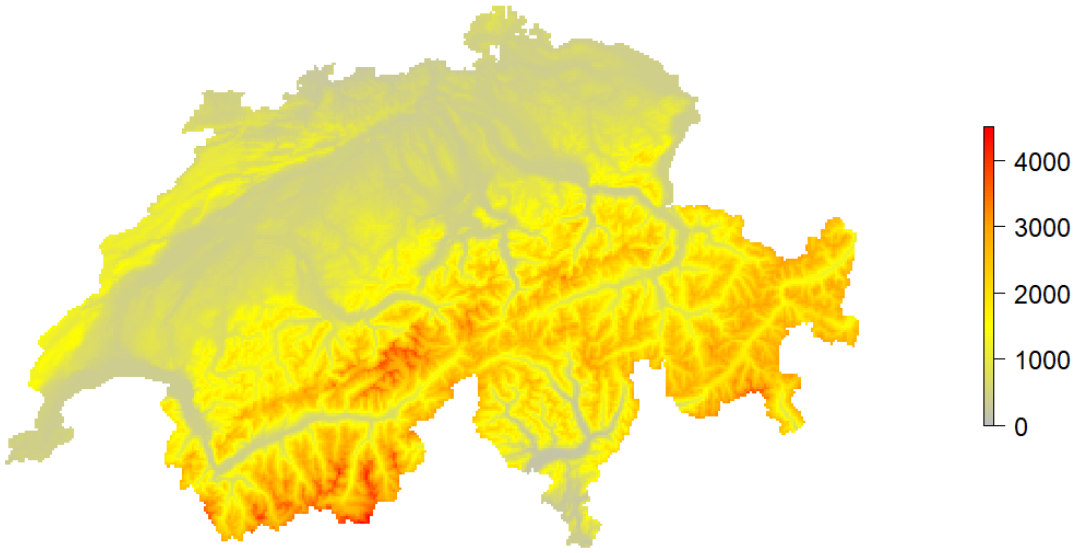


Fig. 19B.2: Map of the elevation covariate in Switzerland. The pixel resolution is 1 km^2 and the domain of Switzerland comprises about 42,000 pixels.

Now, we are going to simulate bullfinch count data for the quadrats in the Swiss breeding bird survey MHB, using the quadrat elevation as a covariate. Note that we pick values of a log-linear regression of expected abundance on elevation that result in an elevational profile of the abundance data that closely resembles similar data collected using the same protocol during the latest Swiss breeding bird atlas (Knaus *et al.* 2018). For data simulation and data analysis, but not for plots, we will work with a transformed variant of the elevation covariate. We will express elevation in kilometric units, which produces more interpretable parameters than what we would get by standardizing this continuous covariate, say, by use of `scale()`. The coefficient for elevation will then quantify the expected change in log-abundance associated with a change in elevation by 1000 metres.

```
# Simulate MHB "look-alike data" for the bullfinch
set.seed(191)

# Pick design constants
nSites <- 267            # Number of quadrats, or sites
nVisits <- 3             # Number of occasions, or visits per site

# Create scaled version of MHB elevation covariate with units of 1 km
mhbElevScaled <- mhbElev/1000
```

We first construct the relationship between elevation and abundance.

```
# Pick values for the regression parameters in expected abundance
alpha.lam <- -3            # Intercept
beta1.lam <- 8.5           # Linear effect of elevation
beta2.lam <- -3.5          # Quadratic effect of elevation

# Compute expected abundance (lambda) for MHB quadrats
lambda <- exp(alpha.lam + beta1.lam * mhbElevScaled + beta2.lam *
mhbElevScaled^2)

# Compute realized abundance (N) for MHB quadrats
N <- rpois(n = nSites, lambda = lambda)
table(N)               # Distribution of abundances across sites
sum(N > 0) / nSites    # Empirical occupancy probability


> table(N)               # Distribution of abundances across sites
N
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 15 16
58 50 32 28 15 14 16 15 12 12  5  3  4  1  1  1

> sum(N > 0) / nSites    # Empirical occupancy proportion
[1] 0.7827715
```

We see that in the simulated MHB bullfinch data set, true abundance per 1 km^2 quadrat varies between 0 and 16 territories, with most values in the range 0–9. Bullfinches occur in 78% of all survey quadrats and are therefore absent in 22% of them.

We make a plot to show the relationships between expected (lambda) and realized bullfinch abundance (*N*) and quadrat elevation (Fig. 19B.3, left). We can compute the value of elevation that is associated with the maximum abundance by computing the vertex of the parabola, or quadratic curve, from the values of coefficients of linear and quadratic elevation above as $-\beta_1 / 2\beta_2$. We find that the true optimum elevation for the bullfinch in Switzerland is 1.21 km.

```
# Figure 19B.3 left
# State process (true states)
par(mfrow = c(1, 3), mar = c(6,6,5,3), cex.lab = 2, cex.axis = 2,
cex.main = 2)
plot(mhbElev, N, main = "Bullfinch abundance", pch = 16, cex = 2,
  col = rgb(0,0,0,0.4), frame = FALSE, xlim = c(0, 3000),
  ylim = c(0, 15), xlab = 'Elevation (m)', ylab = 'Abundance per 1km2')
lines(mhbElev, lambda, lwd = 7, col = rgb(1,0,0,0.4))

# Compute optimal elevation for lambda (in kilometric units !)
(opt.elev.true <- - beta1.lam / ( 2 * beta2.lam) )
```

This concludes our description of the biological process: we have a random process that distributes bullfinches in space, i.e., across sites or quadrats, and we assume that the result of this stochastic process at site i can be approximated by a conditional Poisson distribution with rate parameter λ_i, which itself depends on the average quadrat elevation x_i in a quadratic fashion.

Next, we need to simulate the observation process, i.e., the chance process that generates our observed counts $y_{i,t}$ from true bullfinch abundance N_i, that is, what we will consider the raw data in this chapter. We assume that the observation process is also affected by elevation, because wind speed increases with elevation, and greater wind speeds make bullfinch detection harder. Therefore, we build into the data a negative relationship between detection probability and elevation (Fig. 19B.3 middle).

We note in passing that in the *N*-mixture model, detection probability is defined per individual animal, whereas in the occupancy model in Chapter 19, it refers to the probability to detect *at least one* among the N_i animals or plants present at a site.

```
# Pick values for the 2 regression parameters in detection and get p
alpha.p <- 2
beta.p <- -2
p <- plogis(alpha.p + beta.p * mhbElevScaled)

# Save true parameters into vector
truth <- c(alpha.lam=alpha.lam, beta1.lam=beta1.lam,
  beta2.lam=beta2.lam, alpha.p=alpha.p, beta.p=beta.p)

# Figure 19B.3 middle
# Observation process (characterized by detection probability)
plot(mhbElev, p, xlab = "Elevation (m)",
  ylab = " Detection probability (p)", type = 'l', frame = FALSE,
  xlim = c(0, 3000), lwd = 7, col = rgb(1,0,0,0.4),
  main = "Bullfinch detectability per survey", ylim = c(0, 1))
```

Finally, we simulate three replicated counts, or repeated measurements of abundance, at each site and then look at the data.

```
# Simulate the observation process
C <- array(dim = c(nSites, nVisits))
for(j in 1:nVisits){
  C[,j] <- rbinom(n = nSites, size = N, prob = p)
}
```

We plot the relationship between the observed counts and elevation. In Fig. 19B.3 (right) we see that under the conditions in our data generation process, counts are on average lower than true

abundance, because detection probability is less than 1. In addition, the observed relationship between abundance and site elevation is biased, i.e., the observed counts are not spread out around the red curve representing truth in the right panel in Fig. 19B.3. This is because the same environmental covariate affects both the state process (resulting in abundance *N*) and the observation process (which is represented by the parameter for detection probability, shown in the middle panel). In fact, the expectation of the counts is given by the product of the expected abundance (lambda) and detection probability. To emphasize this, we add the expected count as a black curve in Fig. 19B.3 (right) as well.

```
# Figure 19B.3 right
# Observed counts = 'relative abundance' = 'abundance index'
matplot(mhbElev, C, ylim = c(0, 15), xlab = "Elevation (m)", las = 1,
  ylab = "Counts (C)", main = "Observed bullfinch counts", pch = 16,
  cex = 2, col = rgb(0,0,0,0.4), frame = FALSE, xlim = c(0, 3000))
lines(mhbElev, lambda, type = "l", col = rgb(1,0,0,0.5), lwd = 7)
lines(mhbElev, lambda * p, type = "l", col = "black", lwd = 5)
```

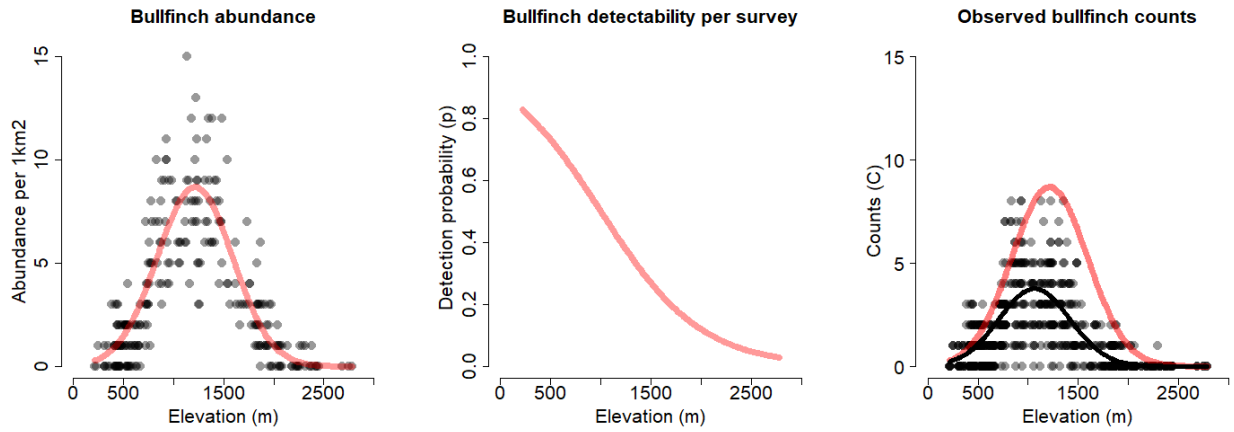


Fig. 19B.3: A simulated system of true abundance and counts at 267 sites where bullfinches are counted. Detection probability is imperfect and depends on the same environmental covariate that also affects abundance, elevation. (Left) The true state: expected abundance (λ ; red line) and the realized abundance (*N*) at the 267 sites. (middle) The relationship between detection probability (*p*) and elevation. (right) Observed counts (i.e., the raw data) at the 267 sites, with the expected true abundance shown as the red line and the expected count (black line) superimposed (red lines left and right are identical).

A conventional analysis would now use some sort of Poisson regression and model the observed counts. One option of such a detection-naïve analysis would be to fit to the replicated counts a Poisson GLMM with site random effects, to accommodate the likely dependence of counts at the same site due to the same underlying *N*. Another common detection-naïve approach is to consider the maximum count only and fit a simple Poisson GLM. Such analyses will model the black bell-shaped cloud in the right panel of Fig. 19B.3, where abundance and detection probability are confounded. Thus, compared to the truth represented by the red line, a conventional detection-naïve analysis will underestimate average abundance and (in our case) will also underestimate the optimum elevation.

It is instructive to inspect the true abundance at each site and compare them with the replicated counts. Our model assumes that we can only make one of the two possible errors: we can only fail to detect individuals, but we never count the same individual twice or make some

other error that leads to false positive errors such as a species misidentification. Thus, in the following comparison we can see that the observed counts can never be greater than the true latent abundance (*N*).

```
# Compare true abundance with the replicated counts
cbind('True state' = N, 'Obs Visit 1' = C[,1], 'Obs Visit 2' = C[,2],
'Obs Visit 3' = C[,3])# Look true state and at three observed states at
each site
```

```
      True state Obs Visit 1 Obs Visit 2 Obs Visit 3
 [1,]          0           0           0           0
 [2,]          0           0           0           0
 [3,]          1           1           1           1
 [4,]          0           0           0           0
 [5,]          1           1           1           1
 [6,]          0           0           0           0
 [7,]          0           0           0           0
 [8,]          0           0           0           0
 [9,]          1           1           1           1
[10,]          0           0           0           0
..........
[141,]         4           2           4           2
[142,]        12           3           5           3
[143,]         8           3           4           3
[144,]         7           5           1           3
[145,]         8           5           4           5
[146,]        10           2           5           3
[147,]         7           2           4           2
[148,]        13           8           5           5
[149,]         8           5           6           2
[150,]        11           3           3           6
..........
[258,]         0           0           0           0
[259,]         1           0           0           0
[260,]         0           0           0           0
[261,]         0           0           0           0
[262,]         0           0           0           0
[263,]         0           0           0           0
[264,]         0           0           0           0
[265,]         0           0           0           0
[266,]         0           0           0           0
[267,]         0           0           0           0
```

We like to emphasize that a species occurrence distribution is fundamentally the same as an abundance distribution, but with much reduced information content: a species occurs at all sites where abundance $N > 0$ (Royle *et al.* 2005; Dorazio, 2007; see also Chapter 20). Hence, any model of abundance is naturally also a model of species distribution in the sense of presence/absence. We believe that it is seldom useful to think of the latter as something separate from abundance.

The next block of code shows us that bullfinches occur in 209 of our 267 simulated MHB quadrats, but that they were detected in only 186 of them. Thus, even though bullfinches were present in the remaining 23 sites, the observers failed to detect them on all three surveys.

```
sum(apply(C, 1, sum) > 0) # Apparent distribution (prop. occupied sites)
sum(N > 0)                # True occupancy
```

```
[1] 186
[1] 209
```

To conclude, we will plot the true abundance species distribution of our simulated bullfinches for the entire modelled domain of Switzerland (Fig. 19B.4). We do this by using our chosen values for the three parameters that govern the expected abundance, `alpha.lam`, `beta1.lam` and `beta2.lam`, along with the elevation values of the Swiss landscape. Adding up the simulated values of expected abundance over all quadrats yields the true Swiss population size of bullfinches in our simulation, which we see is 128,119. This assumes sites are independent, e.g., that no individual lives in more than one site.

```
library(raster)

# Compute true expected abundance (lambda) for all Swiss quadrats
chElevScaled <- chElev / 1000
lambdaCH <- exp(alpha.lam + beta1.lam * chElevScaled + beta2.lam *
chElevScaled^2)

# Inspect quadrat-level expected abundance
summary(lambdaCH)
hist(lambdaCH)       # not shown

# Plot true Swiss bullfinch abundance map (Fig. 19B.4)
par(mfrow = c(1, 1), mar = c(3,4,4,6), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = lambdaCH))
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "True Swiss SDM for the bullfinch (expected abundance)", zlim =
c(0, 9))

# Add up to get Swiss total population size
( Ntot_true <- sum(lambdaCH) )
```

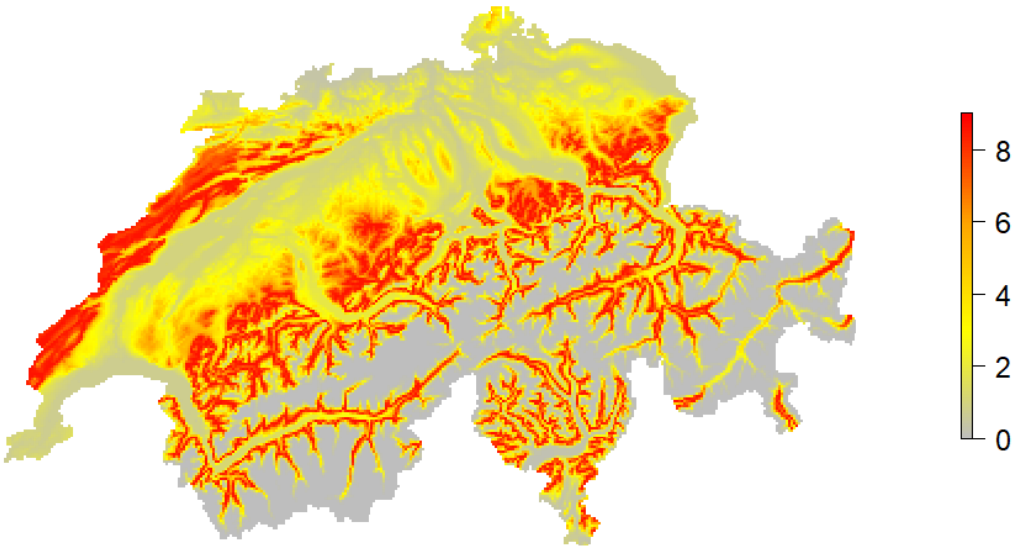


Fig. 19B.4: The true species distribution map of the expected abundance of the Bullfinch (*Pyrrhula pyrrhula*) in Switzerland. The total national population size in our simulation is 128,119 bullfinch territories.

19B.3 A detection-naïve analysis of the maximum count per site

If we were content with ignoring imperfect detection, we could fit a Poisson GLMM (see Chapter 14) to the site-by-visit matrix of bullfinch counts, or a simpler Poisson GLM (Chapter 13) to the vector of the maximum count per site. The maximum count is our best guess at how many bullfinches occurred at each site, and indeed, use of the maximum count per site is a common approach to compress the information in replicated counts. Hence, here is a quick and dirty conventional analysis of the maximum counts assuming a Poisson distribution for this index of abundance. We fit the model with linear and quadratic effects of elevation and make predictions to obtain the elevation profile of the expected counts. We also compute approximate Wald-type prediction intervals.

```
maxC <- apply(C, 1, max)
fm <- glm(maxC ~ mhbElevScaled + I(mhbElevScaled^2), family = poisson)
summary(fm) # not shown
pred.elev <- seq(0.2, 2.5, length.out = 1000)  # Prediction covariate
lpred <- predict(fm, type = 'link', se = TRUE,
    newdata = data.frame(mhbElevScaled = pred.elev))
pred <- exp(lpred$fit)
LCL <- exp(lpred$fit-2*lpred$se)
UCL <- exp(lpred$fit+2*lpred$se)
```

We compare the true number of bullfinch territores in the 267 simulated quadrats and a detection-naïve estimate of this quantity, which is given by the sum of the maximum counts.

```
sum(maxC)   # Detection-naïve estimate of total N in 267 sample quadrats
sum(N)      # True total N in 267 sample quadrats
```

```
> sum(maxC)   # Detection-naïve estimate of total N in 267 sample quads.
[1] 534

> sum(N)      # True total N in 267 sample quadrats
[1] 908
```

We compare the true elevation profile of Swiss bullfinches and the profile estimated in the detection-naïve analysis in a plot (Fig. 19B.4). We also compute the optimal elevation for bullfinch abundance under the detection-naïve model and add that to the plot. We see that in the detection-naïve analysis, we underestimate the optimal elevation for Swiss bullfinches by 96 meters relative to the known truth.

```
# Fig. 19B.5
par(mar = c(5,5,4,2), cex.lab = 1.5, cex.axis = 1.5)
plot(1000*pred.elev, pred, type = 'l', col = rgb(0,0,1, 0.4), lwd = 3,
  xlab = "Elevation (m)", ylab = "Number", xlim = c(0, 3000), ylim =
c(0, 10), main = "Confounding of state and observation processes", frame
= FALSE)
polygon(c(1000*pred.elev, rev(1000*pred.elev)), c(LCL, rev(UCL)), col =
rgb(0,0,1, 0.2), border = NA)
points(mhbElev, maxC, col = rgb(0,0,0, 0.4), pch = 16, cex = 1.3)
lines(mhbElev, lambda, type = "l", col = rgb(1,0,0,0.5), lwd = 5)

(opt.elev2 <- -fm$coef[2] / ( 2 * fm$coef[3]))
(opt.elev2 - opt.elev.true) # Difference to true opt.elev in simulation
abline(v = 1000*opt.elev2, col = rgb(0,0,1,0.4), lwd = 3) # Observed
abline(v = 1000*opt.elev.true, col = rgb(1,0,0,0.4), lwd = 3) # True


> (opt.elev2 <- -fm$coef[2] / ( 2 * fm$coef[3]))
 mhbElev
1.118199

> (opt.elev2 - opt.elev.true) # Difference to true opt.elev in
simulation
    mhbElev
-0.09608642
```

Clearly, the predictions under the detection-naïve analysis yield a biased picture of the elevation profile of bullfinch abundance, since in our simulation bullfinches are easier to detect at lower elevations.

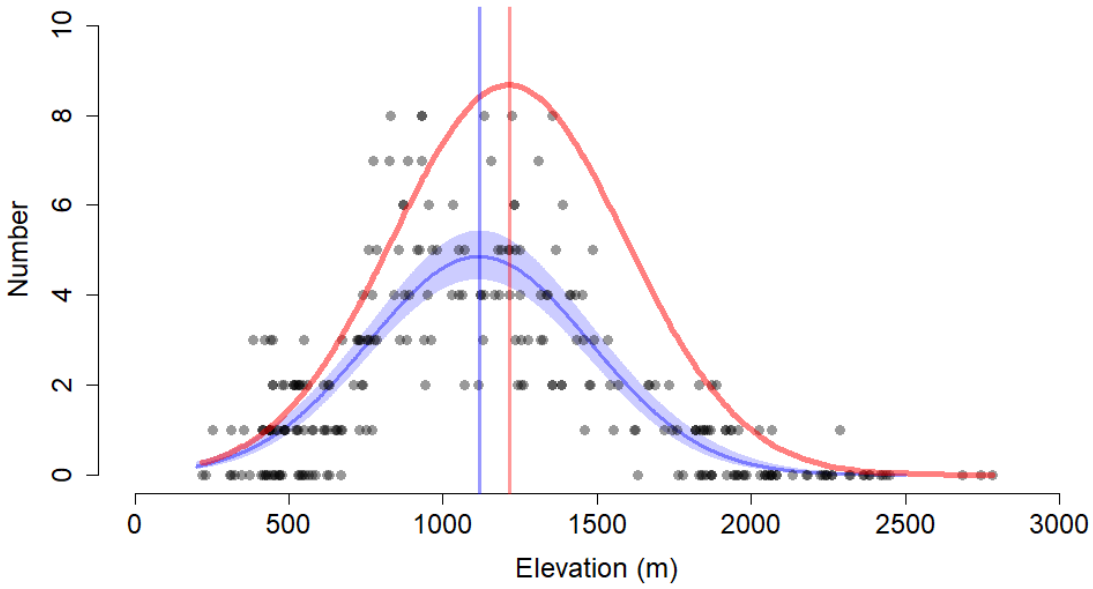


Fig. 19B.5: Relationship between bullfinch abundance and elevation in Switzerland, as inferred by a detection-naïve analysis which does not account for detection probability (expected counts, blue, with 95% CI). The true relationship between expected abundance (lambda) and elevation is shown by the red line. The blue band is an estimate of the black curve in the right panel in Fig. 19B.3.

For illustration, we go on making spatial predictions from the detection-naïve Poisson GLM fitted to the maximum count at each site. We can obtain regional population size estimates from such spatial predictions. Under the assumption that no individual can occur in more than a single quadrat (i.e., that quadrats are independent), we can simply add the predictions over the quadrats in the domain of interest. Below, we add them up over all of Switzerland, to obtain a detection-naïve estimate of the Swiss national population size of bullfinches.

In Fig. 19B.6 we find that the bullfinch occurs most commonly at medium elevation. Ignoring imperfect detection in our analysis of the maximum count per site, we obtain a detection-naïve national population size estimate of 73,651 pairs, which is 43% too low. Note that to obtain a SE or a CI for this estimate, we could use a non-parametric bootstrap; see Section 6.4. in Kéry & Royle (2016).

```
# Make predictions for all Switzerland: prototypical SDM (Fig. 19B.6)
pred.lam_obs <- predict(fm, type = "response",
  newdata = data.frame(mhbElevScaled = chElev/1000))
summary(pred.lam_obs)

par(mfrow = c(1, 1), mar = c(3,4,4,6), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pred.lam_obs))
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Detection-naive Swiss SDM for the bullfinch (expected max
counts)", zlim = c(0, 6))

# Estimated national "relative population size"
( SwissTotalMaxC <- sum(pred.lam_obs) )


> ( SwissTotalMaxC <- sum(pred.lam_obs) )
[1] 73651.16
```

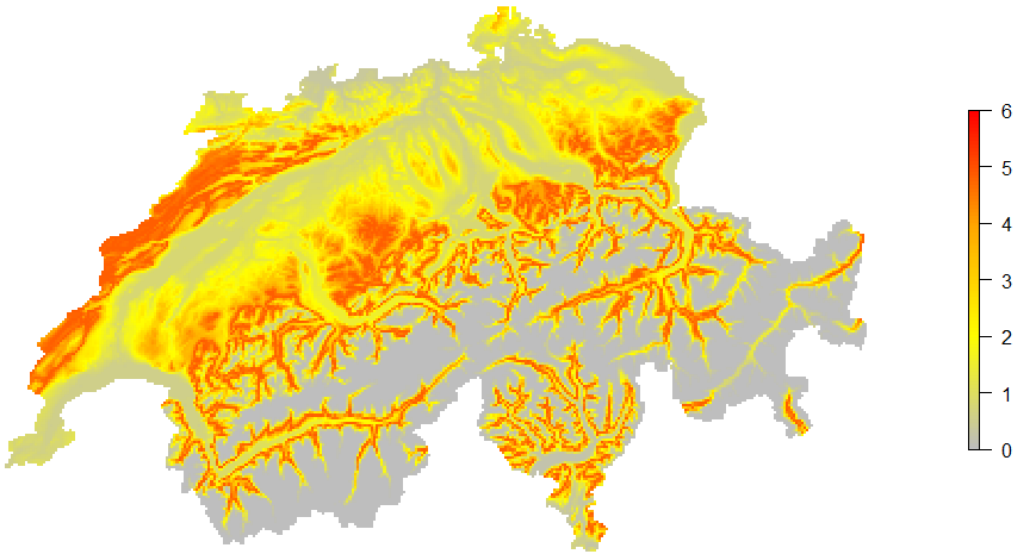


Fig. 19B.6: A detection-naïve species distribution map of the relative abundance (or the expected counts) of the Bullfinch (*Pyrrhula pyrrhula*) in Switzerland. We can add up the quadrat-specific estimates of expected relative abundance over the approx. 42,000 1km^2 quadrats and arrive at a detection-naïve national population size estimate of 73,651 bullfinch territories.

Now, let's see whether we can improve with a hierarchical model that separately estimates the true, latent state (i.e., abundance) and the parameters of the observation process linking the true latent state and the observed counts. That is, we will now deploy all our usual model-fitting engines to fit a binomial *N*-mixture model to the site-by-visit matrix of simulated bullfinch counts.

```
# load required packages
library(ASMbook); library(jagsUI); library(nimble); library(rstan);
library(TMB)
```

19B.4 Likelihood analysis with canned functions in the R package `unmarked`

For our canned function to fit the model using maximum likelihood, we use again the R package `unmarked` (Fiske & Chandler, 2011; Kellner *et al.*, 2023), which has function `pcount()` to fit the binomial *N*-mixture model to data from closed populations. In the first subsection, we will fit the model. In the second subsection, we will make spatial predictions of the expected abundance λ as a function of the estimated values of the involved parameters and the values of elevation in the whole Swiss landscape. These predictions are derived quantities, for which the `predict()` function does the error propagation for us and directly yields prediction standard errors and 95% prediction intervals.

However, sometimes we will want to compute other types of functions of parameters, for which we cannot use an inbuilt `predict()` method. Therefore, in the third subsection, we show two general, non-Bayesian methods for obtaining an uncertainty assessment for a derived quantity. Specifically, our derived quantity in this chapter is optimum elevation, which is a function of two parameters for which we obtain estimates when fitting the model. We will give an illustration of the *delta method* and a simple application of a *parametric bootstrap* to obtain standard errors around the estimate of the optimum elevation. In Chapter 2, we have mentioned the delta method in Section 2.5.5., and we have given full coverage of both the parametric and the non-parametric bootstrap in Section 2.5.4.

19B.4.1 Fitting the model to the simulated bullfinch data set

As with the occupancy model in `unmarked` in Chapter 19, we first need to organize the data by creating a specific type of `unmarkedFrame` data object. This may be confusing at first and so when you start with `unmarked`, it is best to use some template code, e.g., from the help files or from the books covering `unmarked` by Kéry & Royle (2016, 2021). Note that in a real analysis we would want to consider centering or scaling of the elevation covariate, but we omit this here for simplicity, since our analysis turns out to work without this.

```
# Load unmarked, format data and summarize
library(unmarked)
summary( umf <- unmarkedFramePCount(y = C,
  siteCovs = data.frame(elev=mhbElevScaled, elev2=mhbElevScaled ^2)) )

unmarkedFrame Object

267 sites
Maximum number of observations per site: 3
```

```
Mean number of observations per site: 3
Sites with at least one detection: 186

Tabulation of y observations:
  0   1   2   3   4   5   6   7   8
322 185 120  71  39  39  11   8   6

Site-level covariates:
      elev             elev2
 Min.   :0.2193   Min.   :0.04809
 1st Qu.:0.5827   1st Qu.:0.33958
 Median :1.1165   Median :1.24649
 Mean   :1.1875   Mean   :1.82244
 3rd Qu.:1.8187   3rd Qu.:3.30749
 Max.   :2.7807   Max.   :7.73212
```

Next, we use the `pcount` function along with an R formula and our `unmarkedFrame` to fit the *N*-mixture model. As with the occupancy model in `unmarked`, the formula has two parts: the first corresponds to the linear model for detection and the second to the linear model for abundance.

```
out19B.4 <- pcount(~ elev ~ elev + elev2, data = umf)
```

We get a warning from `unmarked` about the value of *K* – what is *K*? Remember that we are estimating a set of random effects (the true abundances *N* at each site) and thus as in Chapters 7, 10, 14, and 17, we need to integrate out these random effects when fitting the model with integrated likelihood. The value of *K* represents the upper bound of this integration. It should be set so that it is much larger than the maximum possible true abundance at any one site. Of course, we don't know this value (that's why we're fitting the model), so we'll have to make an informed guess. If we set *K* too low, we will get incorrect parameter estimates. However, you don't want to make the value of *K* too large either, since the larger the value of *K*, the slower the model will run. `unmarked` sets the value by default at the maximum count + 100. This is likely much larger than necessary for our dataset and indeed for most data sets (Kéry 2018).

To illustrate the effect of the choice of *K* on model results, below we re-fit the model with varying values of *K* and plot the resulting AIC scores. Once *K* is "large enough" we should see the AIC values stabilize.

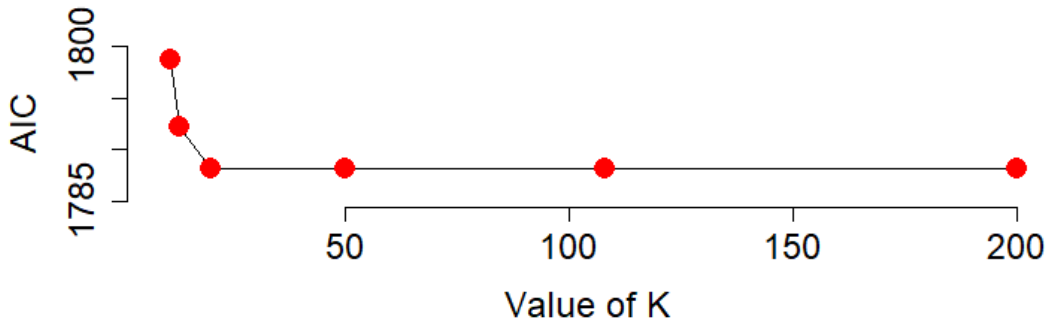
```
# Check for sensitivity of solutions to choice of K
fhm11 <- pcount(~elev ~ elev+elev2, data=umf, K = 11)
fhm13 <- pcount(~elev ~ elev+elev2, data=umf, K = 13)
fhm20 <- pcount(~elev ~ elev+elev2, data=umf, K = 20)
fhm50 <- pcount(~elev ~ elev+elev2, data=umf, K = 50)
fhDef <- pcount(~elev ~ elev+elev2, data=umf)  # With default K = 108
fhm200 <- pcount(~elev ~ elev+elev2, data=umf, K = 200)

# Show only the part where something happens ...
aic <- c(fhm20@AIC, fhm50@AIC, fhDef@AIC, fhm200@AIC)
k <- c(20, 50, 108, 200)
loglik <- sapply(list(fhm20, fhm50, fhDef, fhm200), logLik)

# ... or the whole range of values tried out for K
aic <- c(fhm11@AIC, fhm13@AIC, fhm20@AIC, fhm50@AIC, fhDef@AIC,
fhm200@AIC)
k <- c(11, 13, 20, 50, 108, 200)
loglik <- sapply(list(fhm11, fhm13, fhm20, fhm50, fhDef, fhm200),
logLik)


# Plot both AIC and log-likelihood (Fig. 19B.7)
par(mfrow=c(2,1), mar = c(6,6,4,2), cex.lab = 1.5, cex.axis = 1.5)
plot(k, aic, type='l', ylab="AIC", xlab="Value of K", xlim=c(9, 200),
frame = FALSE, ylim = c(1785, 1800))
points(k, aic, col='red', pch=16, cex = 2)

plot(k, loglik, type='l', ylab="log-likelihood", xlab="Value of K",
     xlim=c(9, 200), frame = FALSE, ylim = c(-895, -888))
points(k, loglik, col='blue', pch=16, cex = 2)
```



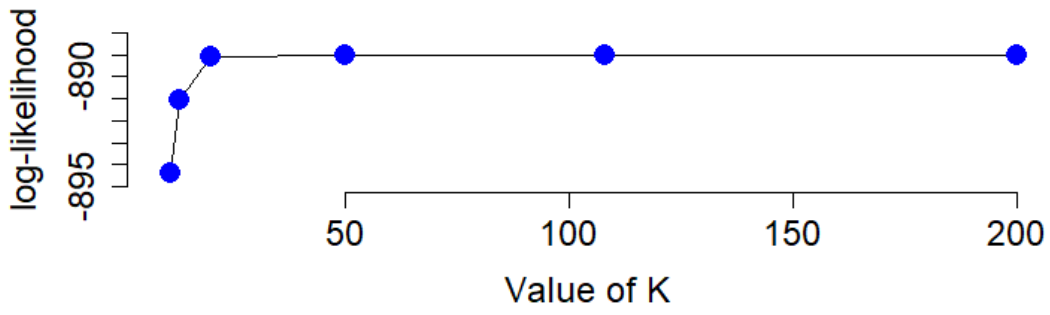


Fig. 19B.7: Assessment of whether the chosen summation limit (*K*) in the numerical evaluation of the integrated likelihood of the *N*-mixture model is sufficient, i.e., does not affect the solutions. We can do this assessment by varying the value of *K* and observing how the AIC or the log-likelihood change. The default *K* in `unmarked` is 100 plus the highest count in the data set.

We see in Figure 19B.7 that choosing a value of *K* below 20 has large impacts on the model estimates, but the default value of (in our case) 108 should be adequate.

Using our original model, we go on to inspect the estimates first and then use the `predict` function to investigate the functional forms between expected abundance (`lambda`) and detection probability (*p*) on the one hand, and elevation on the other (Fig. 19B.8). In contrast to the detection-naive analysis (Fig. 19B.5) we find the inferences from the model that corrects for imperfect detection more satisfactory, albeit with considerable uncertainty in the predictions of `lambda` for higher values of the vegetation covariate.

```
print(out19B.4)
unm_est <- coef(out19B.4)       # Save estimates

Call:
pcount(formula = ~elev ~ elev + elev2, data = umf)

Abundance:
            Estimate    SE     z  P(>|z|)
(Intercept)    -3.55 0.304 -11.7 1.38e-31
elev            9.20 0.594  15.5 3.27e-54
elev2          -3.66 0.320 -11.4 2.35e-30

Detection:
            Estimate    SE     z  P(>|z|)
(Intercept)     2.18 0.352  6.19 5.97e-10
elev           -2.25 0.438 -5.15 2.67e-07

AIC: 1788.085

# Make predictions of abundance and detection(Fig. 19B.8)
state.pred <- predict(out19B.4, type = 'state')
det.pred <- predict(out19B.4, type = 'det')
p.pred <- matrix(det.pred[,1], nrow = nSites, byrow = TRUE) # reformat
p.LCL <- matrix(det.pred[,3], nrow = nSites, byrow = TRUE)  # reformat
p.UCL <- matrix(det.pred[,4], nrow = nSites, byrow = TRUE)  # reformat
```

```
# Plot predicted elevation profiles for state and observation processes
par(mfrow = c(1, 2), mar = c(5,5,4,2), cex.lab = 1.5, cex.axis = 1.5)
# Expected abundance (lambda)
plot(mhbElev, state.pred[,1], xlab = 'Elevation (m)', ylab = 'lambda',
frame = FALSE, col = 'blue', lwd = 3, main = 'State process', type =
'l', ylim = c(0, 15))
lines(mhbElev, lambda, lwd = 3, col = 'red')
polygon(c(mhbElev, rev(mhbElev)), c(state.pred[,3],
rev(state.pred[,4])), col = rgb(0,0,1, 0.2), border = NA)
legend('topleft', legend = c('Estimate', 'Truth'), lwd = 3, col =
c('blue', 'red'), bty = 'n', cex = 1.3)

# Detection
plot(mhbElev, p.pred[,1], xlab = 'Elevation (m)', ylab = 'Detection
prob.', frame = FALSE, col = 'blue', lwd = 3, main = 'Observation
process', type = 'l', ylim = c(0, 1))
lines(mhbElev, p, lwd = 3, col = 'red')
polygon(c(mhbElev, rev(mhbElev)), c(p.LCL[,1], rev(p.UCL[,1])), col =
rgb(0,0,1, 0.2), border = NA)
```

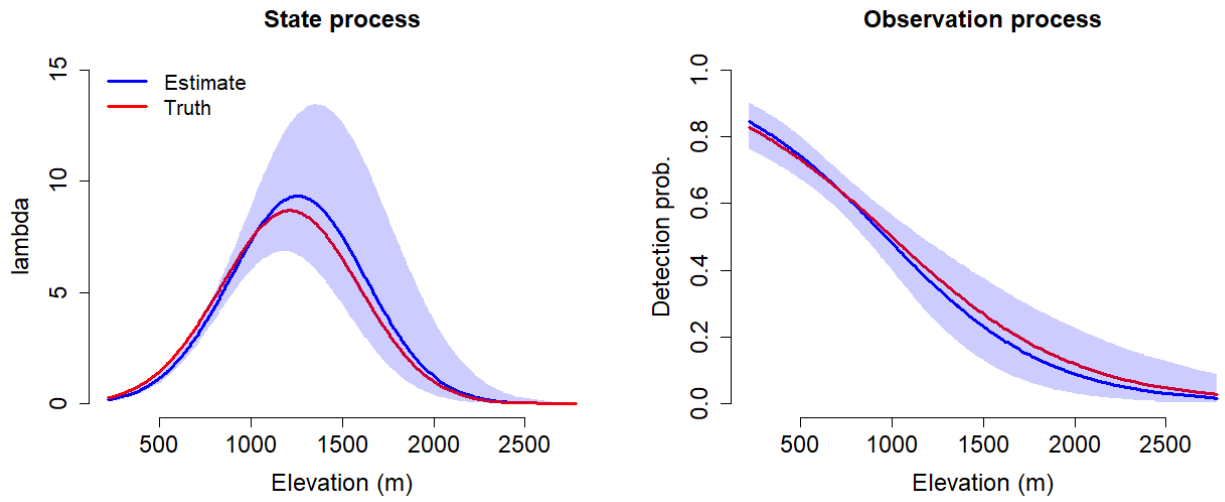


Fig. 19B.8: Comparison of the true relationships with elevation (red) and those estimated from the binomial *N*-mixture model fit with maximum likelihood in the R package `unmarked`.

```
# Compare parameter estimates with truth
comp <- cbind(truth=truth, unmarked=unm_est)
print(comp, 4)
```

```
          truth unmarked
alpha.lam  -3.0   -3.554
beta1.lam   8.5    9.205
beta2.lam  -3.5   -3.661
alpha.p     2.0    2.181
beta.p     -2.0   -2.252
```

```
# Compare estimated total population size in surveyed sample to truth
true_Ntotal <- sum(N)
naive_Ntotal <- sum(apply(C, 1, max))
unm_Ntotal <- sum(bup(ranef(out19B.4)))
comp <- c(truth=true_Ntotal, naive=naive_Ntotal, unmarked=unm_Ntotal)
print(comp, 1)

   truth    naive unmarked
     908      534       979
```

19B.4.2 Spatial prediction of expected abundance

Next, we make spatial predictions on the entire Swiss elevation landscape to compare with the detection-naïve species distribution map in Fig. 19B.6. Spatial prediction is an application of the estimated regression relationship to the environmental values in some real landscape. That is, we can get spatial predictions of the expected bullfinch abundance per 1 km^2 quadrat by plugging the estimates of the three parameters in the abundance model into the equation $\lambda_i = \exp(\alpha + \beta_1 x_i + \beta_2 x_i^2)$ along with the actual values of elevation (linear and squared) in Switzerland. (Note that this is what we get by exponentiating both sides of the linear predictor in the model, $\log(\lambda_i) = \alpha + \beta_1 x_i + \beta_2 x_i^2$.)

Clearly, predictions are a form of derived quantity, or functions of parameters, since they depend (in our case) on the values of three estimated parameters, α, β_1 and β_2. Therefore, the uncertainty of this function will depend on the estimates of these parameters and on the uncertainty in each of them. When we use the function `predict()` on a fitted model object in `unmarked`, we directly obtain the point estimates, along with standard errors and a 95% confidence interval. The point estimates correspond to what we get when we evaluate $\lambda_i = \exp(\alpha + \beta_1 x_i + \beta_2 x_i^2)$ with the MLEs and the supplied values of the covariate *x*, while the SEs and CIs are obtained internally using the delta method; see Section 2.5.5 and 19B.4.3.

```
# Make predictions for all Switzerland
pred.lam <- predict(out19B.4, type = "state",
  newdata = data.frame(elev = chElevScaled, elev2 = chElevScaled^2))
str(pred.lam)
head(pred.lam)
summary(pred.lam)
```

We get one prediction, with SE and confidence limits, for every quadrat in the Swiss landscape for which we supplied values of the elevation covariate in the `newdata` argument. Note that by default, these predictions are on the inverse-link scale, i.e., these are mean abundance estimates, and not log-transformed mean abundance estimates.

```
'data.frame':   42275 obs. of  4 variables:
 $ Predicted: num  0.428 0.428 0.557 0.594 0.48 ...
 $ SE       : num  0.0636 0.0636 0.0751 0.0781 0.0684 ...
 $ lower    : num  0.32 0.32 0.428 0.459 0.363 ...
 $ upper    : num  0.573 0.573 0.726 0.769 0.634 ...

> head(pred.lam)
  Predicted         SE     lower     upper
1 0.4283817 0.06364944 0.3201536 0.5731963
2 0.4283817 0.06364944 0.3201536 0.5731963
```

```
3 0.5571211 0.07512157 0.4277348 0.7256458
4 0.5938605 0.07812907 0.4588801 0.7685456
5 0.4796815 0.06841139 0.3627070 0.6343806
6 0.4796815 0.06841139 0.3627070 0.6343806

> summary(pred.lam)
   Predicted            SE               lower             upper        
 Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   : 0.0000  
 1st Qu.:0.6326   1st Qu.:0.09586   1st Qu.:0.3761   1st Qu.: 0.9771  
 Median :1.7130   Median :0.20225   Median :1.2206   Median : 2.5810  
 Mean   :3.0648   Mean   :0.55442   Mean   :2.2071   Mean   : 4.5285  
 3rd Qu.:5.3069   3rd Qu.:0.92584   3rd Qu.:3.8726   3rd Qu.: 7.8909  
 Max.   :9.3217   Max.   :1.99346   Max.   :6.8537   Max.   :13.4439
```

We can now plot or otherwise visualize or summarize these predictions. First, we plot both the point predictions as well as their SEs and the lower and upper bounds of their 95% CI (Fig. 19B.9). This is the Holy Grail in Chapter 8 of the AHM1 book.

```
# The Holy Grail (Fig. 19B.9)
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))

par(mfrow = c(2, 2), mar = c(3,4,4,6), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pred.lam[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Swiss SDM for the bullfinch\n(estimated true abundance,
lambda)", zlim = c(0, 10))
r2 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pred.lam[,2]))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Prediction SE", zlim = c(0, 2))
r3 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pred.lam[,3]))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Uncertainty (Lower limit of 95% CI)", zlim = c(0, 14))
r4 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pred.lam[,4]))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Uncertainty (Upper of 95% CI)", zlim = c(0, 14))
```

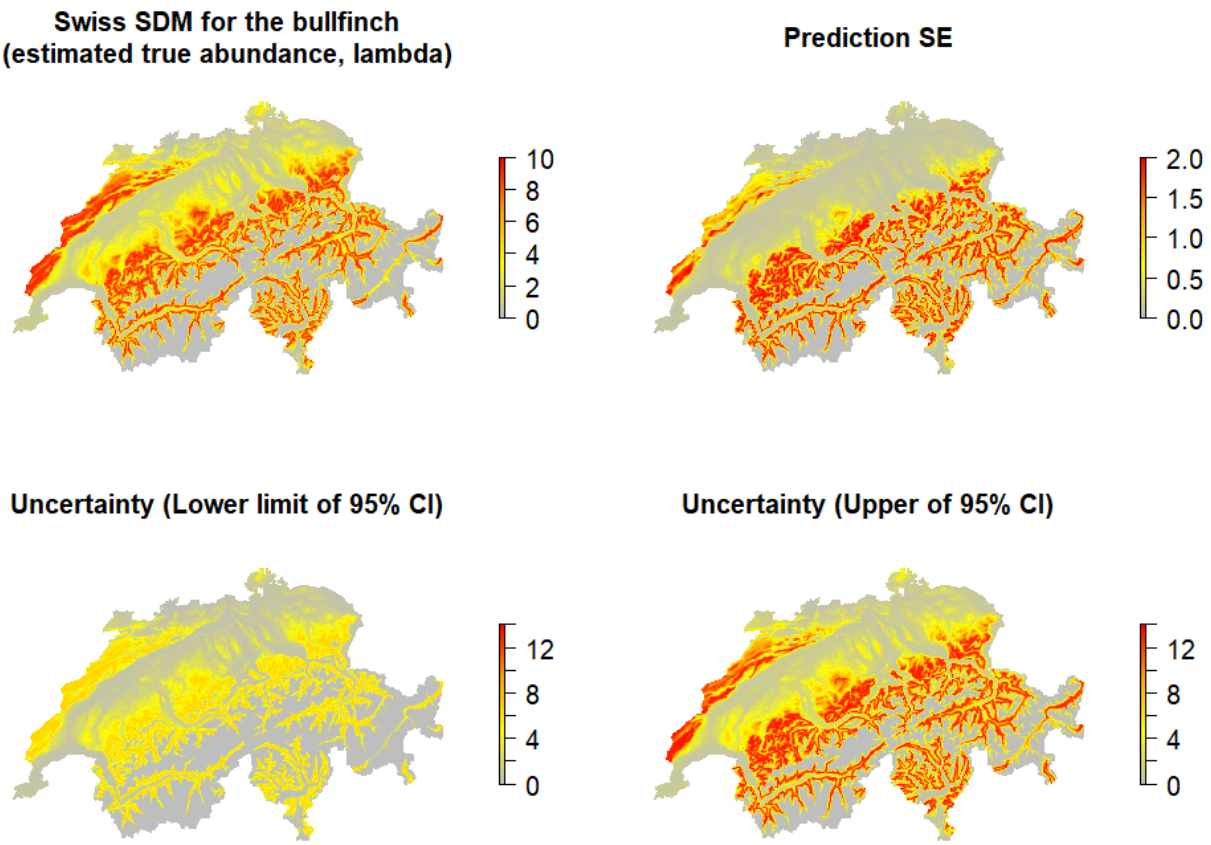


Fig. 19B.9: The Holy Grail for simulated Swiss bullfinches: A species distribution map of the estimated abundance (lambda) in Switzerland (top left). Compare this with the detection-naïve variant of this SDM in Fig. 19b.5. Adding up the map yields a national population size estimate of 129,562 bullfinches. The other maps show the SEs and the lower and upper bounds of a 95% CI.

To obtain regional or national population size estimates, we can simply add up the quadrat-level predictions for the desired region or (here) for the entire country. Note that this always assumes independence of the quadrats, which includes the assumption that no bird can be detected in more than a single quadrat.

```
# Estimated national population size
( SwissTotalN <- sum(pred.lam[,1]) )

# Compare with true national population size
Ntot_true
```

```
> ( SwissTotalN <- sum(pred.lam[,1]) )
[1] 129562.3

> # Compare with true national population size
> Ntot_true
[1] 128119.2
```

This estimate is almost embarrassingly close to the true value. (Disclaimer: You will not always get so good results from an *N*-mixture model. You can easily ascertain this by changing the seeds at the start of this chapter and repeating everything up to here.)

19B.4.3 Computing SEs for a derived quantity using the delta method and the bootstrap

What if we have a derived quantity that is not a prediction of a regression model and thus we cannot use `predict()` to obtain point estimates and uncertainty assessments? The optimum elevation of Swiss bullfinch abundance is such a function. It is the ratio of two regression parameters, or more specifically, $\frac{-\beta_1}{2 \cdot \beta_2}$. To obtain a standard error around this estimate in a frequentist mode of inference, we can use the delta method or a bootstrap. We next give an illustration of both.

```
# Point estimate of optimal elevation (in metres; unmarked solution)
opt.elev.um <- -coef(out19B.4)[2] / (2 * coef(out19B.4)[3])
opt.elev.um
```

```
[1] 1.257289
```

The delta method gives a linear approximation to the standard error of a function of estimated parameters in a model. To apply it, we must either be able to derive it from first principles (for good explanations, see Williams et al., 2002, and Powell, 2007), or else we must find a place where we can look it up for our specific function of interest. In addition, as we will see below, we need the MLEs of the parameters and the variance-covariance matrix of these estimates.

Luckily for us, Powell (2007) gives the equation for the variance, i.e., the squared standard error, of the ratio of two estimated parameters $\hat{\alpha}$ and $\hat{\beta}$, when there is a covariance between them, i.e., when their estimates are correlated (see his equations 15–17).

$$\text{var}(\frac{\hat{\alpha}}{\hat{\beta}}) = (\frac{\hat{\alpha}}{\hat{\beta}})^2 \cdot \left[\frac{\text{var}(\hat{\alpha})}{\hat{\alpha}^2} + \frac{\text{var}(\hat{\beta})}{\hat{\beta}^2} - \frac{2 \cdot \text{cov}(\hat{\alpha}, \hat{\beta})}{\hat{\alpha} \cdot \hat{\beta}} \right]$$

For our application of the optimum elevation of Swiss bullfinch abundance, we can set $\alpha = -\beta_1$ and $\beta = 2 \cdot \beta_2$, and we get this angst-inspiring beast:

$$\text{var}(\frac{-\hat{\beta}_1}{2 \cdot \hat{\beta}_2}) = (\frac{-\hat{\beta}_1}{2 \cdot \hat{\beta}_2})^2 \cdot \left[\frac{\text{var}(\hat{\beta}_1)}{\hat{\beta}_1^2} + \frac{\text{var}(2 \cdot \hat{\beta}_2)}{(2 \cdot \hat{\beta}_2)^2} - \frac{2 \cdot \text{cov}(\hat{\alpha}, \hat{\beta})}{\hat{\beta}_1 \cdot \hat{\beta}_2} \right]$$

Thus, we can see that the variance (i.e., the square of the SE) of the estimate of optimum elevation depends on the point estimates of the coefficients of elevation (linear) and elevation (squared), on their variance, and also on the correlation between these two estimates --- this is the covariance term at the end.

Next, we pull out the required ingredients from the `unmarked` fitted model object. If you're unsure about what the raw output from function `optim()` means, please go back to Chapters 2 and 4 for a refresher.

```
# Get the ingredients for the beast
out19B.4                    # Output from pcount()
str(out19B.4@opt)           # Internal output from optim()
( VC <- solve(out19B.4@opt$hessian) ) # Get VC matrix from Hessian

( b1_hat <- coef(out19B.4)[2]) # Point estimate of elev linear
( b2_hat <- coef(out19B.4)[3]) # ... of elev squared
( var_b1 <- VC[2,2])        # Variance of elev linear
( var_b2 <- VC[3,3])        # ... of elev squared
( cov_b1_b2 <- VC[2,3])     # Covariance between elev linear and squared
```

Finally, we plug them into the equation from above.

```
# Get delta-method approximation of SE of optimum elevation
var_elev.opt <- (-b1_hat / (2*b2_hat))^2 *
                (((2^2 * var_b2) / (2^2 * b2_hat^2)) +
                (var_b1 / b1_hat^2 ) -
                ((2 * cov_b1_b2) / (b1_hat * b2_hat) ) )
sqrt(var_elev.opt)                # SE is square root of variance

> sqrt(var_elev.opt)                # SE is square root of variance
 lam(elev)
0.05350604
```

Excitingly, this seems to work! But let's validate this solution. We will do this by comparing this estimate with that obtained from a parametric bootstrap for the variance estimation of an estimator. We use a simple parametric bootstrap (without refitting of the model) to draw a large number of samples from the sampling distribution of β_1 and β_2 (Fig. 19B.10, left), and compute the desired function for each pair of values (β_1, β_2). In this way, we obtain the sampling distribution of the target function, i.e., of the optimum elevation. Then, we take the standard deviation of that, which provides our parametric bootstrap solution to the problem.

Importantly, we won't draw β_1 and β_2 from two separate normal distributions, but rather from a bivariate normal distribution, which preserves the correlation between the two estimates. This is analogous in a sense to incorporating the covariance term in the delta method formula above. We also plot the empirically derived (i.e., bootstrapped) sampling distribution of the optimum elevation (Fig. 19B.10, right).

```
# Get parametric bootstrap approximation of SE of optimum elevation
library(MASS)               # Load MASS for mvrnorm()
?mvrnorm                    # Check syntax

# Get mean vector and vc matrix for beta1 and beta2
( mu <- c(b1_hat, b2_hat)) # Collect means
( vc <- VC[2:3, 2:3])       # Get relevant part of VC matrix from above

# Draw large number of values from this multivariate normal distribution
boot_beta <- mvrnorm(n = 100000, mu = mu, Sigma = vc)

# Compute the function that gives optimum elevation
opt.elev_boot <- -boot_beta[,1] / (2 * boot_beta[,2])
```

```
# Plot bivariate sampling distribution of beta1 and beta2 and
# sampling distribution of -b1 / (2 x b2)      # Fig. 19B.10
par(mfrow = c(1, 2), mar = c(5,5,5,6), cex.main = 1.5, cex.lab = 1.5,
  cex.axis = 1.5)
plot(boot_beta, asp = 1, xlab = 'beta1', ylab = 'beta2', frame = FALSE,
  pch = 16, col = rgb(0, 0, 0, 0.2), main = 'Bivariate sampling
distribution\n of beta1 and beta2')
hist(1000*opt.elev_boot, breaks = 50, main = 'Sampling distribution\nof
optimum elevation', xlab = 'Elevation (m)', xlim = c(1000, 1600))
```

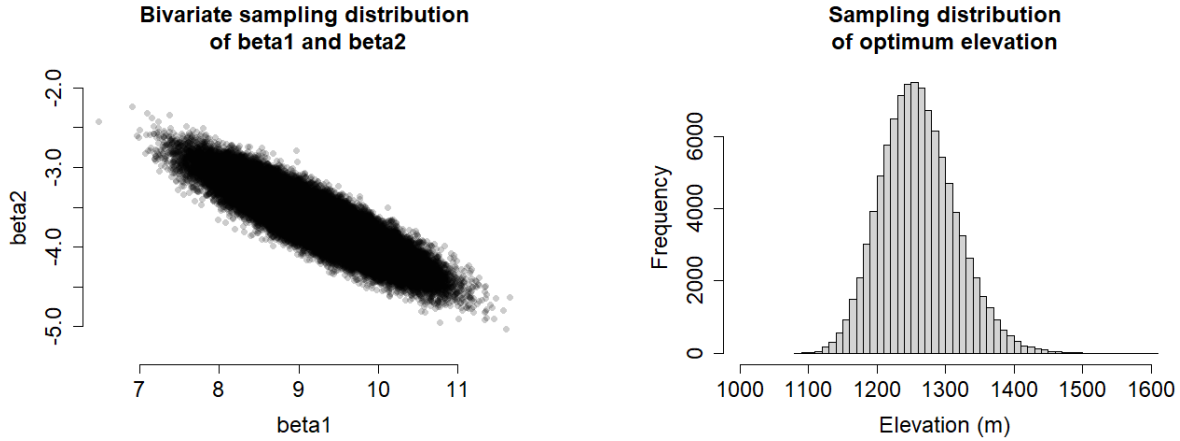


Fig. 19B.10: Empirical sampling distributions (left) of the coefficients of elevation linear (β_1) and elevation squared (β_2) and (right) of the frequentist estimator of the optimum elevation for the abundance of simulated Swiss bullfinches, i.e., of the function $-\beta_1 / (2 \cdot \beta_2)$.

We compute the 95% CIs based on the bootstrapped sampling distribution of the optimum elevation. And finally, we compare the delta-method and the parametric bootstrap solutions and find them very similar (note that this will not always be the case).

```
# Get parametric-bootstrapped SE and CI of optimum elevation
# SE and CIs are SD and 0.025/0.975th percentiles
SE_opt.elev <- sd(opt.elev_boot)
( CI_opt.elev <- quantile(opt.elev_boot, c(0.025, 0.975)))

# Compare two non-Bayesian solutions to the SE of optimum elevation
comp <- c(sqrt(var_elev.opt), SE_opt.elev)
names(comp) <- c("Delta method", "Parametric Bootstrap")
print(comp, 4)
```

```
> ( CI_opt.elev <- quantile(opt.elev_boot, c(0.025, 0.975)))
    2.5%    97.5%
1.163774 1.377985

> print(comp, 4)
        Delta method Parametric Bootstrap
             0.05351              0.05474
```

In the next section, we will compare these SE estimates with a Bayesian solution based on posterior draws.

19B.5 Bayesian analysis with JAGS

Next, the Bayesian solution with JAGS. We will also add into the code computation of the optimum elevation of the expected abundance of our simulated bullfinches in Switzerland.

```
# Bundle and summarize data
elev2 <- mhbElevScaled^2          # squared elevation
str(dataList <- list(C=C, elev = mhbElevScaled, elev2 = elev2,
  nSites = nSites, nVisits = nVisits) )
```

```
List of 5
 $ C      : int [1:267, 1:3] 0 0 1 0 1 0 0 0 1 0 ...
 $ elev   : num [1:267] 0.219 0.231 0.253 0.311 0.311 ...
 $ elev2  : num [1:267] 0.0481 0.0532 0.064 0.0965 0.097 ...
 $ nSites : num 267
 $ nVisits: num 3
```

In our model we also add code to assess goodness-of-fit using a posterior predictive check for a Chi-squared discrepancy measure. Note that we know that the coefficients for elevation in the abundance part of the model are quite large. Therefore, we use wide Gaussian priors in the analysis, since we want our priors to be vague to minimize the amount of information that they contribute towards the estimation. Of course, this is not something that we could know with a data set that was not simulated. However, also with a real-world data set we could either compare different priors (i.e., conduct a prior sensitivity analysis) or compare the Bayesian point estimates with the MLEs for different priors.

```
# Write JAGS model file
cat(file="model19B.5.txt", "
model {
# Priors
mean.lam ~ dunif(0, 1)
alpha.lam <- log(mean.lam)
beta1.lam ~ dnorm(0, 0.001)
beta2.lam ~ dnorm(0, 0.001)
mean.p ~ dunif(0, 1)
alpha.p <- logit(mean.p)
beta.p ~ dnorm(0, 0.001)

# Likelihood
# Biological model for true abundance
for (i in 1:nSites) {             # Loop over sites
  N[i] ~ dpois(lambda[i])
  log(lambda[i]) <- alpha.lam + beta1.lam * elev[i] + beta2.lam *
elev2[i]
}

# Observation model for replicated counts
for (i in 1:nSites) {             # Loop over sites
  for (t in 1:nVisits) {          # Loop over all n observations
    C[i,t] ~ dbin(p[i,t], N[i])
    logit(p[i,t]) <- alpha.p + beta.p * elev[i]
  }
}

# Derived quantities
totalN <- sum(N[])   # Estimate total population size across all sites
opt.elev <- -beta1.lam / (2 * beta2.lam)  # Optimum elevation

# Assess model fit using Chi-squared discrepancy
for (i in 1:nSites) {             # Loop over sites
  for (t in 1:nVisits) {          # Loop over all n observations
    # Compute fit statistic for observed data
    eval[i,t] <- p[i,t]*N[i]
    E[i,t] <- pow((C[i,t] - eval[i,t]),2) / (eval[i,t] + 0.01)# Avoid
div. by 0 !
    # Generate replicate data and compute fit stats for them
    C.new[i,t] ~ dbin(p[i,t], N[i])
    E.new[i,t] <- pow((C.new[i,t] - eval[i,t]),2) / (eval[i,t] + 0.01)
  }
}
fit <- sum(E[,])
fit.new <- sum(E.new[,])
}
```

As for the occupancy model in Chapter 19, adequate starting values for the latent abundance states, i.e., the latent variables N_i, are essential. We use the maximum count at each site as a guess of what N might be and add 1 to avoid zeros, which sometimes cause initialization problems.

```
# Function to generate starting values
Nst <- apply(C, 1, max) + 1
inits <- function(){list(N = Nst, mean.lam=runif(1, 0, 1),
beta1.lam=rnorm(1, 0, 1), beta2.lam=rnorm(1, 0, 1), mean.p=runif(1),
beta.p=rnorm(1, 0, 1))}
```

For convenience, we add in the list of parameters to estimate the intercepts of abundance and detection in both their link-scale and their natural-scale variants.

```
# Parameters to estimate
params <- c("alpha.lam", "beta1.lam", "beta2.lam", "alpha.p", "beta.p",
"mean.lam", "mean.p", "totalN", "fit", "fit.new", "opt.elev", "N")
```

N-mixture models are one of the harder models in this book to fit in practice and we need long chains. With this model you now get the first hint at what Bayesian modeling often feels like 'in practice', i.e., for the analyses on which many of the papers that you read are based one often needs a lot of patience.

```
# MCMC settings
na <- 10000;  ni <- 150000;  nb <- 50000; nc <- 4; nt <- 100
# na <- 50;  ni <- 1100;  nb <- 100; nc <- 4; nt <- 2        # test

# Call JAGS (ART 11 min)
out19B.5 <- jags(dataList, inits, params, "model19B.5.txt", n.iter = ni,
  n.burnin = nb, n.chains = nc, n.thin = nt, n.adapt = na,
  parallel = TRUE)
```

We first check whether this model run has converged by looking at the traceplots and by checking out whether any parameter has a value of the Rhat (Brooks-Gelman-Rubin) statistic that is greater than our rule of thumb of 1.1.

```
# Traceplots (Fig. 19B.11)
jagsUI::traceplot(out19B.5)      # All parameters
jagsUI::traceplot(out19B.5, c("alpha.lam", "beta1.lam", "beta2.lam",
"alpha.p", "beta.p", "opt.elev", "totalN"), layout = c(3, 3)) # Key
parameters only

# Check out the Rhat values
which(out19B.5$summary[,8] > 1.1)          # anybody not converged?
out19B.5$summary[which(out19B.5$summary[,8] > 1.1) ,8] # how bad?

# Look at traceplot for parameter that has not converged
# jagsUI::traceplot(out19B.5, c("N[2]"), layout = c(2, 2))
```

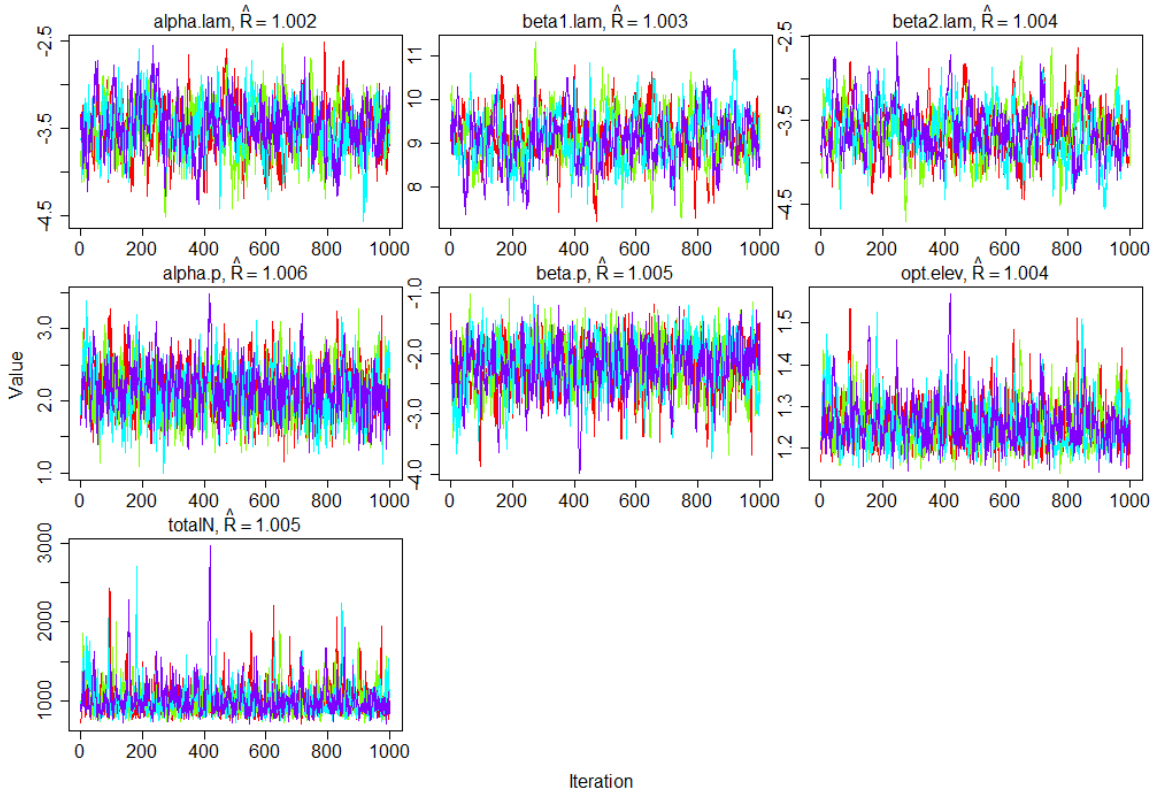


Fig. 19B.11: Traceplots for some of the key parameters in the *N*-mixture model fit to the simulated Swiss bullfinches.

We find that our MCMC settings are good enough to achieve convergence for all main parameters in the model. Depending on the run, the chains for one or two latent variables (or discrete random effects, *N*) may not have quite converged, but in a hierarchical model, we will have to worry most about convergence of the main structural parameters in the upper levels of the hierarchy. For these, convergence is good enough in our case, so we're happy.

Next, we check the goodness-of-fit of the model using a visual posterior predictive check, along with a Bayesian *p*-value, and find fit to be satisfactory (Fig. 19B.12) --- again, not a surprise for our simulated data.

```
# Compute Bayesian p-value and print
( bpv <- mean(out19B.5$sims.list$fit.new > out19B.5$sims.list$fit) )

# Fig. 19B.12
par(mar = c(6,6,5,4), cex.axis = 1.3, cex.lab = 1.3, cex.main = 1.3)
plot(out19B.5$sims.list$fit, out19B.5$sims.list$fit.new,
  main = paste("Bayesian p-value:", round(bpv, 4)),
  xlab = "Discrepancy for actual data set",
  ylab = "Discrepancy for perfect data sets",
  pch= 16, cex = 1.2, col = rgb(0,0,0,0.2), frame = FALSE)
abline(0,1, lwd = 2, col = "black")
```

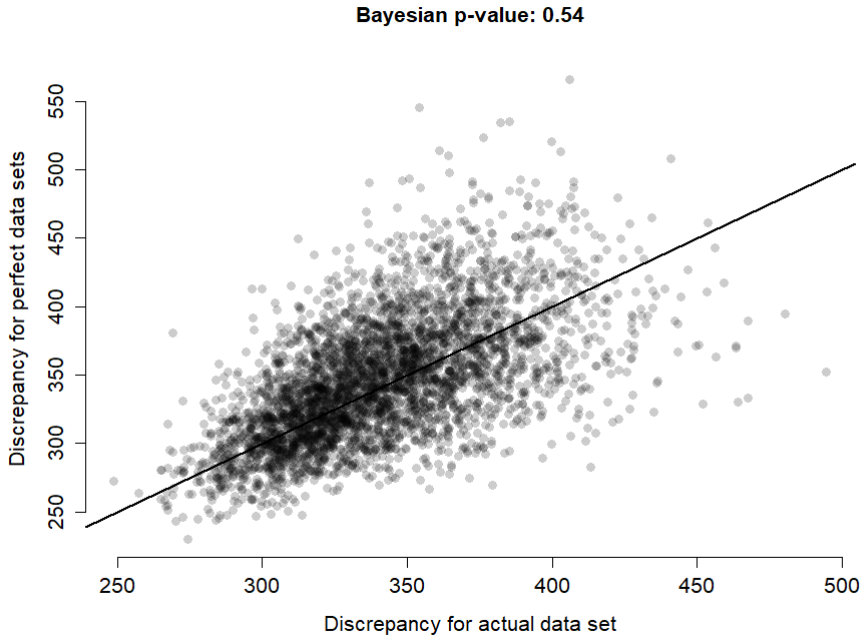


Fig. 19B.12: Posterior predictive check of the binomial N-mixture model using a Chi-squared discrepancy.

The graphical check and the Bayesian *p*-value both indicate an adequate model for our data set. Hence, we inspect the parameter estimates and then compare them with truth in the data-generating process and with the estimates produced by the other engines so far.

```
print(out19B.5, 3)
```

```
              mean      sd     2.5%      50%    97.5% overlap0 f  Rhat n.eff
alpha.lam   -3.498   0.291   -4.082   -3.490   -2.922    FALSE 1 1.002   971
beta1.lam    9.129   0.560    8.013    9.125   10.221    FALSE 1 1.003   785
beta2.lam   -3.635   0.296   -4.192   -3.648   -3.008    FALSE 1 1.004   643
alpha.p      2.127   0.344    1.485    2.116    2.842    FALSE 1 1.006   400
beta.p      -2.212   0.426   -3.120   -2.181   -1.479    FALSE 1 1.005   479
mean.lam     0.032   0.009    0.017    0.031    0.054    FALSE 1 1.003   924
mean.p       0.889   0.033    0.815    0.892    0.945    FALSE 1 1.008   365
totalN    1012.839 203.256  778.000  965.000 1524.050    FALSE 1 1.005  1062
fit        340.727  32.968  285.632  337.663  411.511    FALSE 1 1.002   931
fit.new    346.017  44.915  273.356  341.703  446.164    FALSE 1 1.001  1682
opt.elev     1.259   0.055    1.177    1.251    1.395    FALSE 1 1.004   665
N[1]         0.001   0.027    0.000    0.000    0.000     TRUE 1 1.051  4000
N[2]         0.001   0.027    0.000    0.000    0.000     TRUE 1 1.292  1333
N[3]         1.004   0.067    1.000    1.000    1.000    FALSE 1 1.022  4000
......
```

Fig. 19B.13 gives a graphical comparison between the parameter estimates under the *N*-mixture model and the values of the associated data-generating parameters. It shows again that the model does a good job at estimating abundance. In the bottom-right panel, we note that the posterior distribution of `totalN` is strongly skewed to the right. This is quite typical for parameters that are bounded on one side, such as population sizes, variances, or standard deviations. For them, our typical choice of the posterior mean as a Bayesian point estimator is not ideal, and the mode or the median should be used instead. Therefore, here and further down in this chapter, we will use the posterior median as a point estimator for population size parameters.

We see that with truth being 908 bullfinch territories, our estimate of `totalN` (965 territories) appears quite good when compared with the sum of the max counts across all 267 sites, which is 534.

```
par(mfrow = c(3,2), mar = c(5,5,4,3), cex.lab = 1.5, cex.axis = 1.5)
hist(out19B.5$sims.list$alpha.lam, col = "grey", main = "alpha.lam",
xlab = "", breaks = 50)
abline(v = alpha.lam, lwd = 3, col = "red")
hist(out19B.5$sims.list$beta1.lam, col = "grey", main = "beta1.lam",
xlab = "", breaks = 50)
abline(v = beta1.lam, lwd = 3, col = "red")
hist(out19B.5$sims.list$beta2.lam, col = "grey", main = "beta2.lam",
xlab = "", breaks = 50)
abline(v = beta2.lam, lwd = 3, col = "red")
hist(out19B.5$sims.list$alpha.p, col = "grey", main = "alpha.p", xlab =
"", breaks = 50)
abline(v = alpha.p, lwd = 3, col = "red")
hist(out19B.5$sims.list$beta.p, col = "grey", main = "beta.p", xlab =
"", breaks = 50)
abline(v = beta.p, lwd = 3, col = "red")
hist(out19B.5$sims.list$totalN, col = "grey", , main = "Total N", xlab =
"", breaks = 50)
abline(v = sum(N), lwd = 3, col = "red")
```

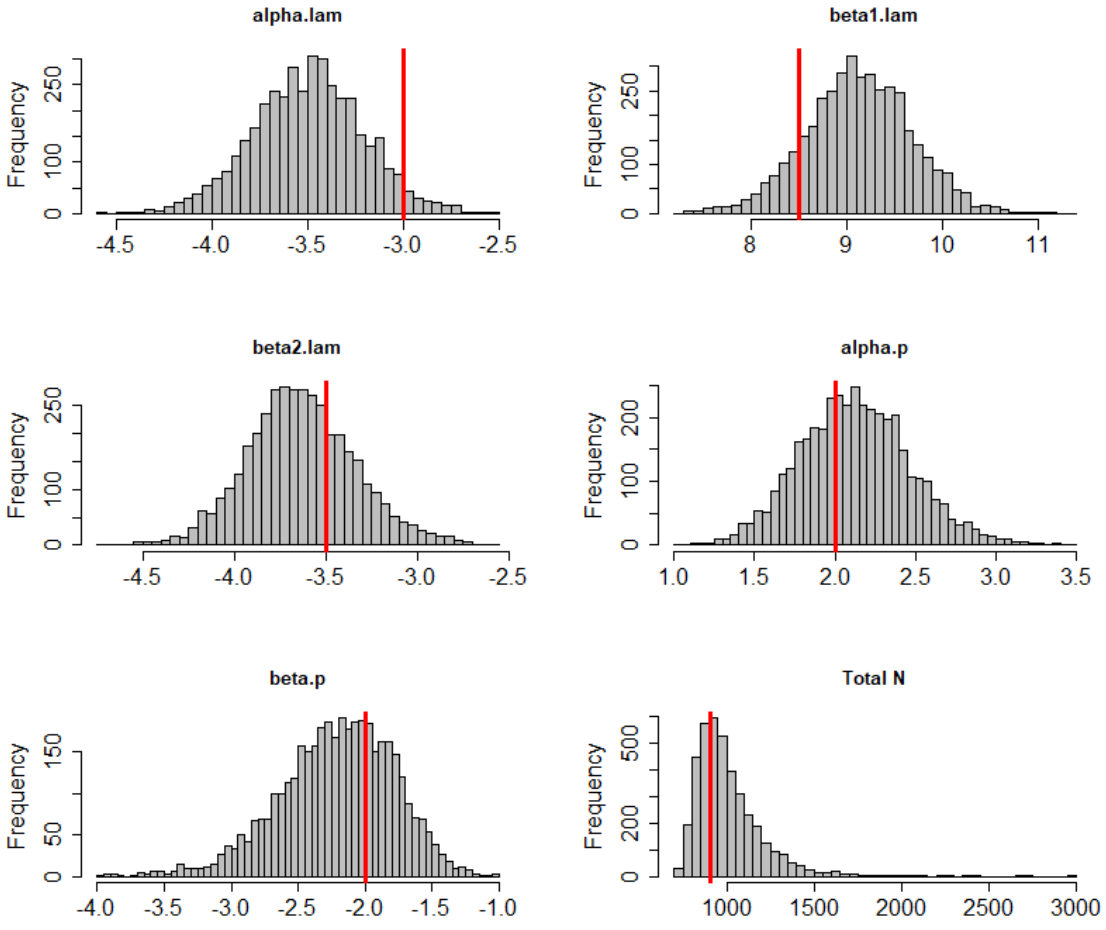


Fig. 19B.13: Comparison of estimates under the binomial *N*-mixture model (posterior distributions) and truth in the data-generating algorithm (red line) for six estimands.

We next illustrate a few further inferences that can be made under the model by looking at more posterior distributions. In Fig. 19B.13 we have seen those for some primary parameters of the model. One of the most interesting things in the *N*-mixture model is perhaps that site-specific abundance estimates, i.e., N_i, can easily be obtained. Let's have a look at these estimates of local abundance for a sample of these sites (Fig. 19B.14). You can browse through all posteriors of *N* using the following code (note to RStudio users: this may break for your variety of R). We see again highly skewed posterior distributions, for which the posterior median or mode should be used as a point estimator, rather than the posterior mean.

```
par(mfrow = c(2,3), mar = c(5,5,5,3), cex.main = 1.5, cex.axis = 1.5,
cex.lab = 1.5, ask = TRUE)
  # for(i in 1:nSites){      # Do this for plots of all sites
  for(i in 151:156){         # Do this to produce Fig. 19B.14

  hist(out19B.5$sims.list$N[,i], col = "grey",
    xlim = range(c(max(C[i,]), out19B.5$sims.list$N[,i])),
    main = paste("Site", i), xlab = "Population size N", freq = FALSE)
  abline(v = Nst[i]-1, lwd = 3, col = "black")
  abline(v = N[i], lwd = 3, lty = 2, col = "red")
}
```

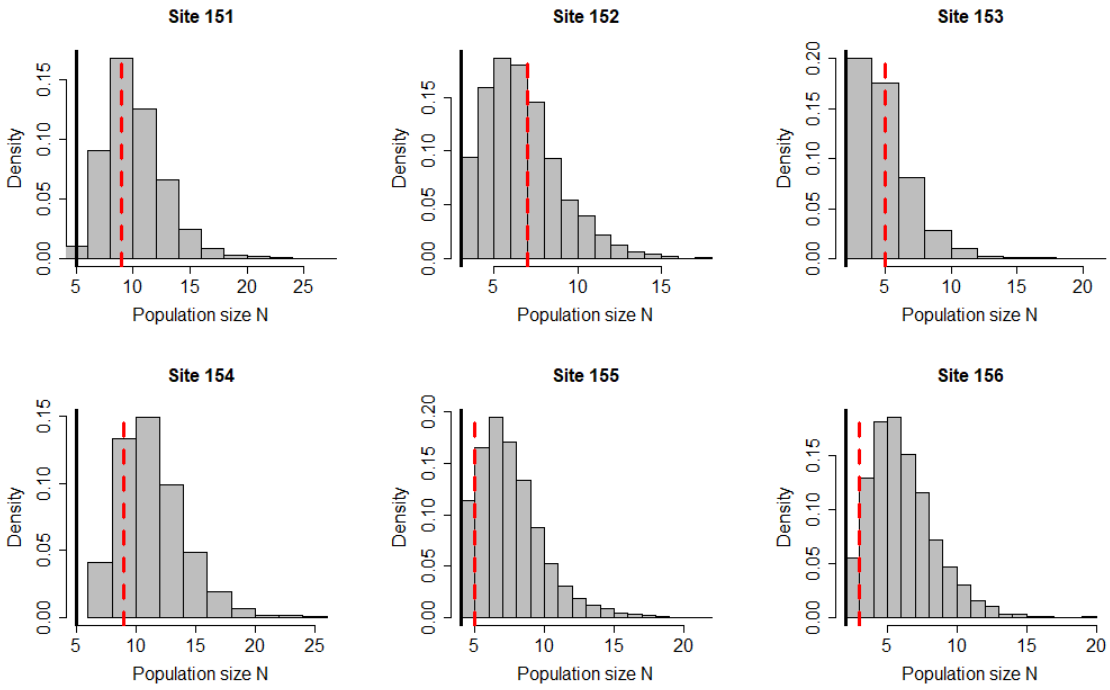


Fig. 19B.14: Comparison of estimates of local population size (N_i) under the *N*-mixture model (posterior distributions) at a sample of six sites. The maximum count at each side is denoted by the black line and the true population size in the simulated data by the dashed red line.

Then, in Fig. 19B.15 we show a comparison of the relationship between bullfinch abundance and elevation using a detection-naïve analysis and under the *N*-mixture model.

```
# Fig. 19B.15
par(mar = c(5,5,5,3), cex.main = 1.5, cex.axis = 1.5, cex.lab = 1.5)
plot(mhbElev, N, main = "", xlab = "Elevation (m)", ylab = "Abundance",
las = 1, ylim = c(0, max(N)), pch = 16, col = rgb(1,0,0,0.5), cex = 1.3,
frame = FALSE)
lines(mhbElev, lambda, col = rgb(1,0,0,0.5), lwd = 5)

Nmix.pred <- exp(out19B.5$mean$alpha.lam + out19B.5$mean$beta1.lam *
mhbElevScaled + out19B.5$mean$beta2.lam * mhbElevScaled^2)
lines(mhbElev, Nmix.pred, type = "l", col = rgb(0,0,1,0.5), lwd = 5)
lines(1000*pred.elev, pred, col = rgb(0,0,0,0.5), lwd = 5)
legend("topright", lwd = 5, col = c(rgb(0,0,1,0.5), rgb(1,0,0,0.5),
rgb(0,0,0,0.5)), legend = c("Nmix", "True", "Detection-naive"))
```

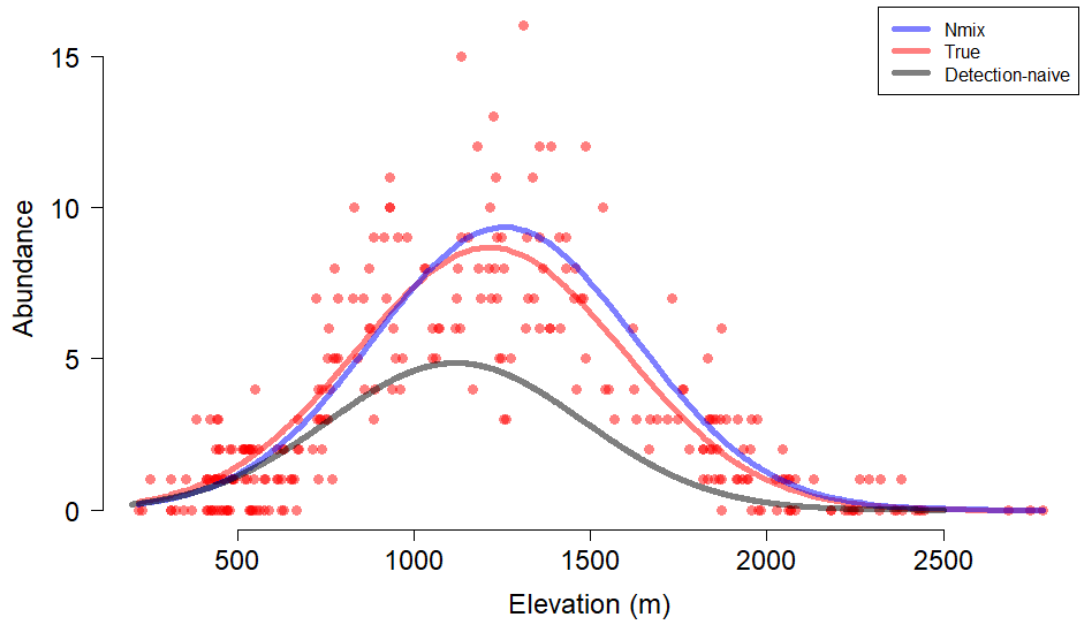


Fig. 19B.15: Comparison of the estimated abundance-elevation relationship in Swiss bullfinches under a detection-naïve approach that ignores imperfect detection (grey) with that under the binomial *N*-mixture model (blue). Truth is shown in red: circles are the realized abundances at each site (N_i) and the line shows the expected abundance.

Comparing the MLEs of the main parameters in the model from `unmarked` with the posterior means we see a very good numerical agreement.

```
# Compare likelihood with Bayesian estimates and with truth
# Parameter estimates
jags_est <- unlist(out19B.5$mean)[1:5]
comp <- cbind(truth = truth, unmarked = unm_est, JAGS = jags_est)
print(comp, 3)
```

```
          truth unmarked  JAGS
alpha.lam  -3.0    -3.55 -3.50
beta1.lam   8.5     9.20  9.13
beta2.lam  -3.5    -3.66 -3.63
```

```
alpha.p      2.0      2.18  2.13
beta.p      -2.0     -2.25 -2.21
```

Concerning the optimum elevation of bullfinch abundance in Switzerland, the detection-naïve analysis yields an underestimate by 96 metres, while the estimates from `unmarked` and JAGS are very similar and overestimate this value by about 45 metres.

```
# Point estimates of optimal elevation (in metres)
comp <- cbind('truth' = 1000*opt.elev.true, 'p-naive' = 1000*opt.elev2,
  'unmarked' = 1000*opt.elev.um, 'JAGS' = 1000*out19B.5$mean$opt.elev)
print(comp, 3)
```

```
        truth p-naive unmarked JAGS
mhbElev  1214    1118     1257 1259
```

In Section 19B.4.3, we have obtained estimated standard errors around this estimate by the delta method and a parametric bootstrap, i.e., by two frequentist methods. With Bayesian posterior inference the posterior SD is the analogous quantity. We find that they are very similar.

```
# Compare three solutions to the SE of optimum elevation
comp <- c(sqrt(var_elev.opt), SE_opt.elev, out19B.5$sd$opt.elev)
names(comp) <- c("Delta method", "Parametric Bootstrap", "Posterior SD")
print(comp, 4)
```

```
        Delta method Parametric Bootstrap         Posterior SD
             0.05351              0.05476              0.05514
```

For a final comparison between a detection-naïve analysis, and a frequentist and a Bayesian analysis using the *N*-mixture model, we look at the estimates of the total population size.

```
# Compare total population size in sampled quadrats to truth
jags_Ntotal <- out19B.5$q50$totalN
comp <- c(truth = true_Ntotal, naive = naive_Ntotal, unmarked =
unm_Ntotal, JAGS = jags_Ntotal)
print(comp, 2)
```

```
   truth    naive unmarked     JAGS
     908      534      979      965
```

We emphasize that many of these comparisons are correct in showing general patterns. For instance, detection-naïve analyses will always underestimate abundance when $p < 1$, while frequentist and Bayesian inferences for the *N*-mixture model will usually lead to estimates that are numerically very similar. However, we have inspected only a single data set and hence, to make more general statements of this kind, you would have to conduct simulations with many data sets (e.g., 20, 100 or 1000). Typically, in such a simulation-based assessment, you would also vary many of the settings of the data simulation, such as sample sizes and parameter values.

Continuing, we will now use the JAGS results to plot spatial predictions of the expected bullfinch abundance (i.e., `lambda`) in Switzerland, to yield a species distribution map. As for the analysis of the model using maximum likelihood from `unmarked`, we will plot both point estimates and uncertainty assessments (posterior SD's and 95% credible intervals), for a Bayesian variant of the Holy Grail for our simulated Swiss bullfinches (Fig. 19B.15). Note also our continued use of the median instead of the mean as a Bayesian point estimator for a skewed posterior.

```
# Make predictions for all Switzerland for a prototypical SDM
# Prelims
samps <- out19B.5$sims.list     # Grab all posterior samples
str(samps)                      # Look at overview
nsamp <- length(samps$alpha.lam)# Sample size

# Create array to hold predictions for all of Switzerland
pred.lam2 <- array(NA, dim = c(nsamp, length(chElev)))

# Fill the array using MCMC draws and Swiss elevation data
for(i in 1:nsamp){
  if(i %% 50 == 0) cat(paste("Predicting for MCMC draw", i, "\n"))
  pred.lam2[i,] <- exp(samps$alpha.lam[i] + samps$beta1.lam[i] *
chElevScaled + samps$beta2.lam[i] * chElevScaled^2)
}

# Compute posterior medians, sds and 95%CRIs
pm <- apply(pred.lam2, 2, median)    # Posterior medians
psd <- apply(pred.lam2, 2, sd)       # Posterior sd
lcl <- apply(pred.lam2, 2, function(x) quantile(x, 0.025)) # Lower CRL
ucl <- apply(pred.lam2, 2, function(x) quantile(x, 0.975)) # Upper CRL


# The Bayesian version of the Holy Grail (see Chapter 8 in AHM1 book)
#   (Fig. 19B.15)
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))
par(mfrow = c(2, 2), mar = c(3,4,4,6), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = pm))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Swiss SDM for the bullfinch\n(estimated true abundance,
lambda)", zlim = c(0, 15))
r2 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = psd))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Uncertainty (posterior sd of lambda)", zlim = c(0, 2.5))
r3 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = lcl))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Uncertainty (Lower credible limit)", zlim = c(0, 15))
r4 <- rasterFromXYZ(data.frame(x = ch$x, y = ch$y, z = ucl))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE,
  main = "Uncertainty (Upper credible limit)", zlim = c(0, 15))
```

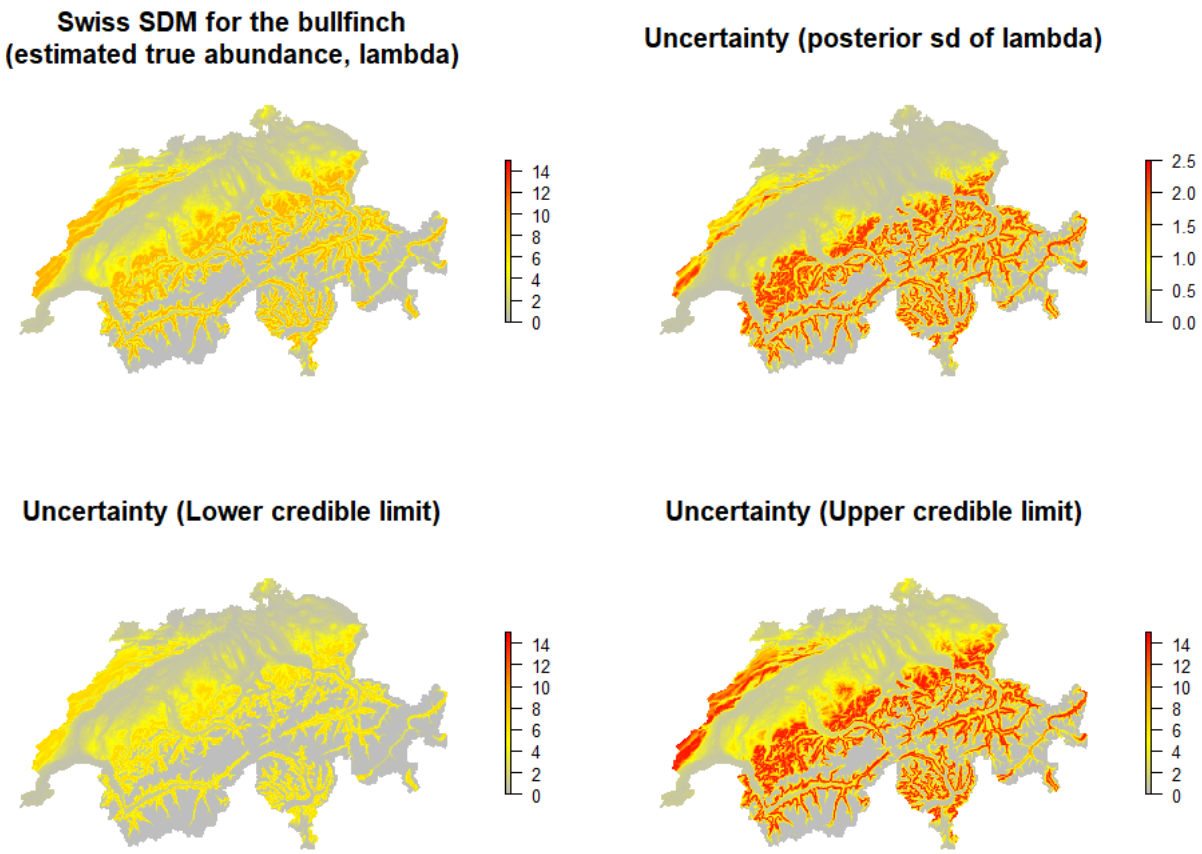


Fig. 19B.16: The Bayesian Holy Grail for simulated Swiss bullfinches: An abundance-based species distribution map of Swiss bullfinches (top left), along with three maps that depict the uncertainty in these estimates: posterior SD of the 1km^2-wise predictions (top right), and the lower and upper 95% credible limits of these predictions (bottom left and right, respectively).

For these maps, we have summarized the prediction matrix `pred.lam2` by kilometric pixel. In contrast, to obtain the posterior distribution of the Swiss national population size of simulated bullfinches, we summarize the same matrix by MCMC sample. This gives us the posterior distribution of the national population size of simulated bullfinches in Switzerland (Fig. 19B.17). From this we can obtain the usual summaries for a point estimate and uncertainty assessments. We find that there are about 129,000 bullfinch territories in Switzerland, with a posterior sd of roughly 25,000 and a 95% credible interval of about 103,000–197,000.

```
# Compute estimated Swiss population size, with posterior sd and CRI
post_Ntot <- apply(pred.lam2, 1, sum)

# Plot full posterior distribution (Fig. 19B.17)
hist(post_Ntot, breaks = 40, col = "gold", main = "Swiss national
population size of simulated bullfinches (Posterior distribution)", xlab
= "Number of territories", ylab = "Density", freq = FALSE)
abline(v = Ntot_true, col = "red", lwd = 4)
abline(v = median(post_Ntot), col = "blue", lwd = 4, lty = 2)

# Compute median, sd and CRI
(pm_Ntot <- median(post_Ntot))
(psd_Ntot <- sd(post_Ntot))
(CRI_Ntot <- quantile(post_Ntot, c(0.025, 0.975)))
```

```
> (pm_Ntot <- median(post_Ntot))
[1] 128649.3

> (psd_Ntot <- sd(post_Ntot))
[1] 25388.22

> (CRI_Ntot <- quantile(post_Ntot, c(0.025, 0.975)))
    2.5%    97.5%
103239.3 196986.5
```

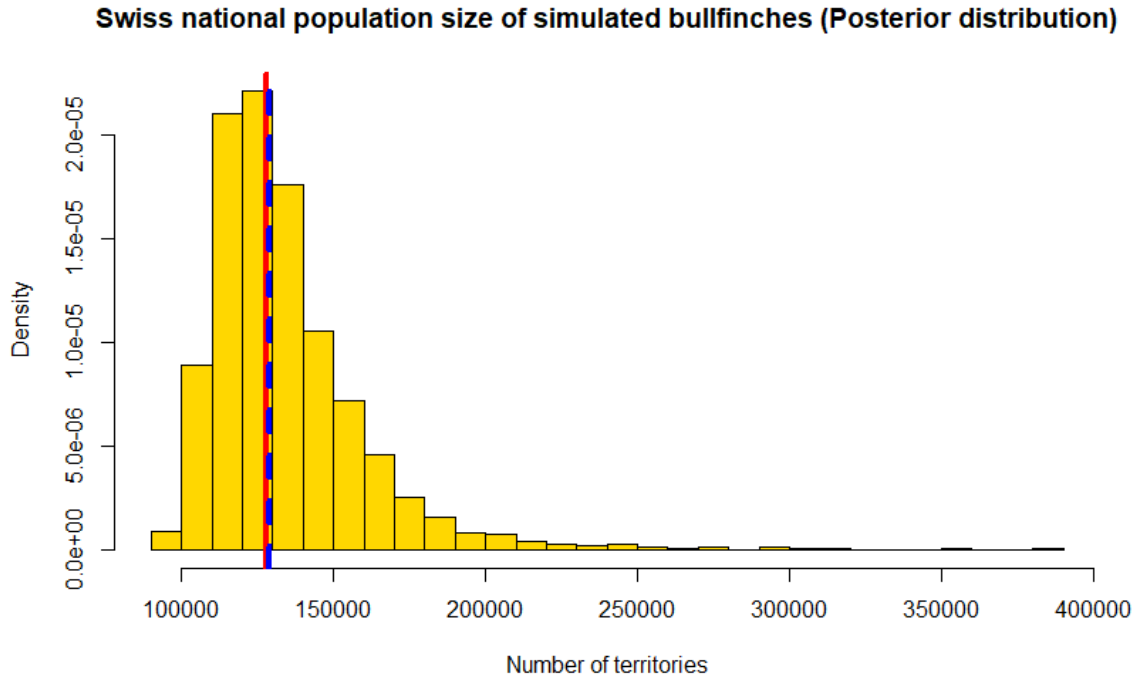


Fig. 19B.17: Posterior distribution of the total population size of simulated bullfinches in Switzerland. This is such a glorious result that we choose a golden color for the bars. Red and blue lines show the true value in the simulated data and the posterior median, respectively.

19B.6 Bayesian analysis with NIMBLE

As always, we can use the JAGS model with virtually no change to fit the same model in NIMBLE. However, we drop the part in the model that computes the posterior predictive check. Initial runs suggested we needed to run longer Markov chains for NIMBLE in this case, so we choose MCMC settings about twice as heavy as for JAGS above.

```
# Bundle and summarize data
elev2 <- mhbElevScaled^2          # squared elevation
str(dataList <- list(C=C, elev = mhbElevScaled, elev2 = elev2,
  nSites = nSites, nVisits = nVisits) )
```

Here is the model written in the BUGS language for NIMBLE.

```
# Write NIMBLE model file
model19B.6 <- nimbleCode( {

# Priors
mean.lam ~ dunif(0, 1)
alpha.lam <- log(mean.lam)
beta1.lam ~ dnorm(0, 0.001)
beta2.lam ~ dnorm(0, 0.001)
mean.p ~ dunif(0, 1)
alpha.p <- logit(mean.p)
beta.p ~ dnorm(0, 0.001)

# Likelihood
# Biological model for true abundance
for (i in 1:nSites) {             # Loop over sites
  N[i] ~ dpois(lambda[i])
  log(lambda[i]) <- alpha.lam + beta1.lam * elev[i] + beta2.lam *
elev2[i]
}

# Observation model for replicated counts
for (i in 1:nSites) {             # Loop over sites
  for (t in 1:nVisits) {          # Loop over all n observations
    C[i,t] ~ dbin(p[i,t], N[i])
    logit(p[i,t]) <- alpha.p + beta.p * elev[i]
  }
}

# Derived quantities
totalN <- sum(N[1:nSites]) # Estimate total pop. size across all sites
opt.elev <- -beta1.lam / (2 * beta2.lam)  # Optimum elevation

} )
```

```
# Inits
Nst <- apply(C, 1, max) + 1
inits <- function(){list(N = Nst, mean.lam=runif(1, 0, 1),
beta1.lam=rnorm(1, 0, 1), beta2.lam=rnorm(1, 0, 1), mean.p=runif(1),
beta.p=rnorm(1, 0, 1))}

# Parameters monitored: same as before
params <- c("alpha.lam", "beta1.lam", "beta2.lam", "alpha.p", "beta.p",
"mean.lam", "mean.p", "totalN", "opt.elev", "N")
params <- c("alpha.lam", "beta1.lam", "beta2.lam", "alpha.p", "beta.p",
"mean.lam", "mean.p", "totalN", "opt.elev")

# MCMC settings
# Number of samples returned is floor((niter-nburnin)/thin)
# ni <- 150000  ;  nb <- 50000  ; nc <- 4  ; nt <- 100  # Like JAGS
ni <- 300000  ;  nb <- 200000  ; nc <- 4  ; nt <- 200   # safer


# Call NIMBLE (ART 38 min), check convergence and summarize posteriors
system.time( out19B.6 <-
    nimbleMCMC(code = model19B.6,
    constants = dataList,
    inits = inits, monitors = params,
    nburnin = nb, niter = ni, thin = nt, nchains = nc,
    samplesAsCodaMCMC = TRUE)    )
par(mfrow=c(2,2), ask = TRUE); coda::traceplot(out19B.6)
nsum <- nimble_summary(out19B.6, params)
print(as.matrix(nsum), 4)          # summary, not shown
nimble_est <- nsum[1:5,1]          # Save estimates
nimble_Ntotal <- nsum[8,4]         # Posterior median
```

19B.7 Bayesian analysis with Stan

To fit the model in Stan we unfortunately cannot use the same intuitive, hierarchical construction of the likelihood as for JAGS and NIMBLE. This is due to the inability of the software to directly deal with discrete random effects, i.e., the latent abundance states of *N*. Hence, as for the occupancy model in Chapter 19, we need to work with the marginal, or integrated, likelihood, where the latent abundance states are eliminated from the likelihood by summation (Berger *et al.*, 1999; Joseph 2020; Yackulic *et al.*, 2020). This makes the implementation of the model harder to understand for nonstatisticians, but it may be still worthwhile to learn this, since marginalization in hierarchical model may speed up computations in general (Yackulic *et al.*, 2020). We note also that both `unmarked` (Section 19B.4) and `ubms` (see next section) implement the *N*-mixture model in this manner, although this is 'under the hood', and so you don't get to see this.

To start building the integrated likelihood, here's the likelihood for one site *i* for a particular value of the true abundance N_i.

$$L_i(\lambda_i, \boldsymbol{p}_i|\boldsymbol{y}_i, T) = \text{dpois}(N_i|\lambda_i) \cdot \prod_{t=1}^{T} \text{dbinom}(y_{i,t}|N_i, p_{i,t})$$

The likelihood for mean abundance λ_i and the vector of survey-specific detection probabilities $\boldsymbol{p}_i$, given the vector of *T* repeated counts $\boldsymbol{y}_i$, is the product of the likelihoods of the Poisson abundance and binomial detection submodels.

Since we don't know the value of the true abundance N_i, we have to integrate over all possible values that it can take. Since N_i is discrete, we can do this by defining the set of all possible values for N_i and calculating the likelihood above for each value, then summing up these likelihoods (i.e. we do integration by summation). So how can we define the set of all possible N_i? The minimum value in this set should be the maximum observed count in $\boldsymbol{y}_i$, since under the assumption of no false positives we know there are at least that many animals present. The maximum should in principle be infinity, but in practice we will approximate that by a large number. That is, we choose some value well above the largest realistic value, exactly as with the choice of *K* in the `unmarked` section above. Here we'll choose *K* to be the maximum count + 20. This results in the following likelihood for site i:

$$L_i(\lambda_i, \boldsymbol{p}_i | \boldsymbol{y}_i, T) = \sum_{k=\max(\boldsymbol{y}_i)}^{K} \left(\text{dpois}(k|\lambda_i) \cdot \prod_{t=1}^{T} \text{dbinom}(y_{i,t}|k, p_{i,t}) \right)$$

The total likelihood for all sites is the product of all the site-specific likelihoods L_i, or more commonly, the sum of the log-likelihoods.

Our dataset for Stan is similar to the one for JAGS, but we also add the summation limits for evaluating the integrals in the likelihood contribution from each site, which we call `Kmin` (lower) and `K` (upper).

```
# Bundle and summarize data
# Build data list (a bit different from JAGS/Nimble)
# Minimum and maximum possible abundance at each site
Kmin <- maxC                # Maximum count at each site
K = max(C) + 20             # Over max count plus some 'buffer'
dataList <- list(C=C, nSites=nSites, nVisits=nVisits,
  elev=mhbElevScaled, elev2= mhbElevScaled ^2, Kmin=Kmin, K=K)
str(dataList)
```

```
List of 7
 $ C      : int [1:267, 1:3] 0 0 1 0 1 0 0 0 1 0 ...
 $ nSites : num 267
 $ nVisits: num 3
 $ elev   : num [1:267] 0.219 0.231 0.253 0.311 0.311 ...
 $ elev2  : num [1:267] 0.0481 0.0532 0.064 0.0965 0.097 ...
 $ Kmin   : int [1:267] 0 0 1 0 1 0 0 0 1 0 ...
 $ K      : num 28
```

Most of the Stan model looks similar to the JAGS code. At the end of the `model` section, we translate the likelihood math above into Stan code, calculate the total log-likelihood across sites, and supply the result to the special Stan `target` as in Chapter 19.

```
# Write Stan model
cat(file="model19B_7.stan", "
data{
  int nSites;
  int nVisits;
  int C[nSites, nVisits];
  vector[nSites] elev;
  vector[nSites] elev2;
  int K;
  int Kmin[nSites];
}

parameters{
  real alpha_lam;
  real beta1_lam;
  real beta2_lam;
  real alpha_p;
  real beta_p;
}

transformed parameters{
  vector[nSites] log_lambda;
  vector[nSites] logit_p;
  for (i in 1:nSites){
    log_lambda[i] = alpha_lam + beta1_lam * elev[i] + beta2_lam *
elev2[i];
    logit_p[i] = alpha_p + beta_p * elev[i];
  }
}

model{
  vector[nSites] lik; //likelihood for each site

  //Priors
  alpha_lam ~ normal(0, 2);
  beta1_lam ~ normal(0, 100);
  beta2_lam ~ normal(0, 100);
  alpha_p ~ normal(0, 10);
  beta_p ~ normal(0, 100);

  for (i in 1:nSites){
    lik[i] = 0;
    for (k in Kmin[i]:K){
      lik[i] += exp(poisson_log_lpmf(k | log_lambda[i]) +
                    binomial_logit_lpmf(C[i,1:nVisits] | k,
logit_p[i]));
    }
    target += log(lik[i]);
  }
}

generated quantities {
  real opt_elev;
  opt_elev = -beta1_lam / (2 * beta2_lam);
}
")
```

```
# Parameters to estimate
params <- c("alpha_lam", "beta1_lam", "beta2_lam", "alpha_p",
  "beta_p", "opt_elev")

# HMC settings
ni <- 1500   ;   nb <- 500   ;  nc <- 4   ;  nt <- 1

# Call STAN (ART 18 min), assess convergence and print results table
options(mc.cores = parallel::detectCores()-1) # Run Stan in parallel
system.time(
out19B.7 <- stan(file = "model19B_7.stan", data = dataList,
  pars = params, warmup = nb, iter = ni, chains = nc, thin = nt) )
rstan::traceplot(out19B.7)        # not shown, but see next paragraph
print(out19B.7, dig = 3)          # not shown
stan_est <- summary(out19B.7)$summary[1:5,1]   # Save estimates
```

At least in our implementation of the model, Stan takes longer than JAGS to run, but it is faster than our implementation of it in NIMBLE. On the other hand, looking at the trace plots (not shown here), we see that the posterior samples are much less correlated than those produced by JAGS or NIMBLE. That is, they effectively contain more information. Thus, the fairest and most meaningful comparison of the efficiency of an MCMC algorithm is not the time it takes for producing a certain number of MCMC samples, but rather the effective sample size per unit of time (Ponisio *et al.*, 2020). And if we computed this, it might well be that Stan is the "fastest" among the three Bayesian engines for our example.

As before (e.g., in Chapter 19), when we fit a hierarchical model with an integrated likelihood, we must do additional calculations to recover estimates of the random effects. Below, we show one approach to calculating an estimate of the total abundance using the Stan output. Similar calculations will be necessary for this for the DIY-MLE and TMB engines below.

```
# Get posterior of sumN

# Get posterior of parameters (beta vector)
stan_beta_post <- as.matrix(out19B.7)[,1:5]

# Function that takes posterior of beta and generates posterior of sumN
get_post_sumN <- function(Beta_samples, K){
  nsims <- nrow(Beta_samples)
  sumN_sims <- rep(NA, nsims)
  for (i in 1:nsims){
    beta <- Beta_samples[i,]
    lambda <- exp(beta[1] + beta[2] * mhbElevScaled + beta[3] *
mhbElevScaled^2)
    p <- plogis(beta[4] + beta[5] * mhbElevScaled)

    N <- rep(NA, nSites)
    for (n in 1:nSites){
      Kvals <- max(C[n,]):K
      Kprob <- rep(NA, length(Kvals))
```

```
      for (k in 1:length(Kvals)){
        pLam <- dpois(Kvals[k], lambda[n])
        pP <- 1
        for (j in 1:nVisits){
          pP <- pP * dbinom(C[n,j], Kvals[k], p[n])
        }
        Kprob[k] <- pLam * pP
      }
      Kprob <- Kprob / sum(Kprob)
      N[n] <- sample(Kvals, 1, prob=Kprob)
    }
    sumN_sims[i] <- sum(N)
  }
  sumN_sims
}

# Bootstrap that function to get CIs as well and produce plot
system.time(     # ART 210 sec
   post_sumN <- get_post_sumN(stan_beta_post, K=30)
   )
stan_Ntotal <- median(post_sumN)

# Draw figure (Fig. 19B.18)
hist(post_sumN, main="Posterior of sumN from Stan", breaks = 40)
abline(v=stan_Ntotal, col='blue', lwd = 5)
abline(v=true_Ntotal, col='red', lwd = 5)
legend("topright", lty=1, col=c('red','blue'),
legend=c('truth','estimate'), lwd = 5, cex = 1.2, bty = 'n')
```

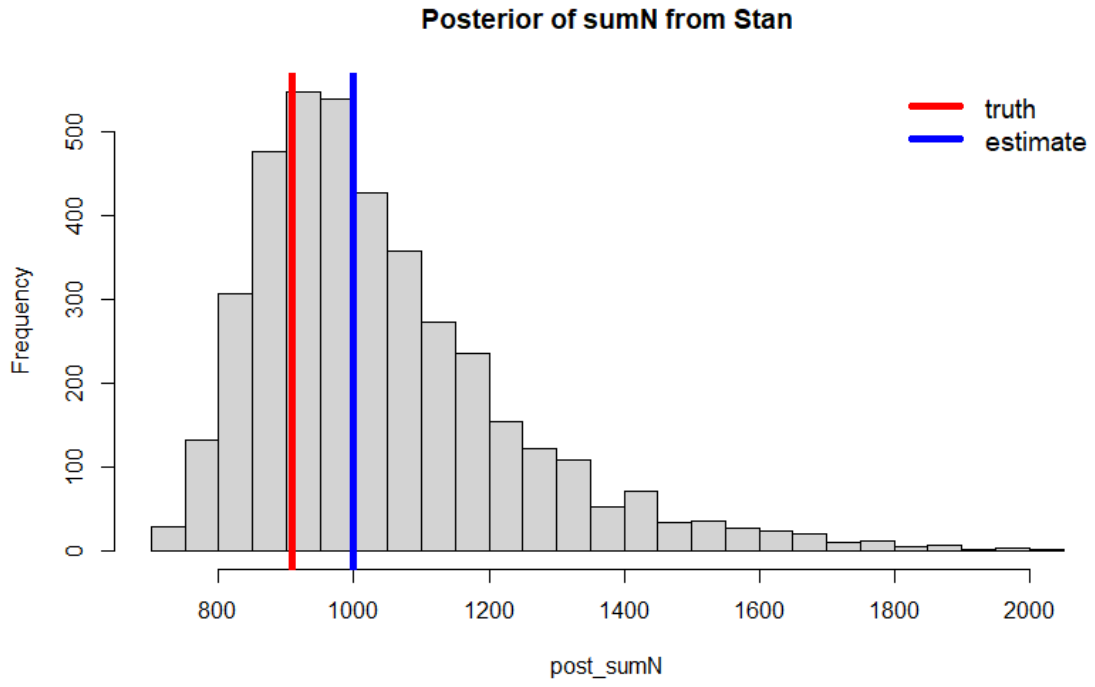


Fig. 19B.18: Posterior distribution of the total number of simulated Swiss bullfinches living in the 267 sample quadrats. Red and blue lines show the true value and the posterior median, respectively.

19B.8 Bayesian analysis with canned functions in the R package ubms

We encountered the package ubms already in Chapter 19. We have seen that it is a wrapper for Stan and has virtually the same input and syntax as unmarked. Therefore, if you know unmarked, then the transition to ubms will be extremely easy. We note that fitting this *N*-mixture model in ubms, using Stan under the hood, takes much longer than what we might expect based for instance on our experience with occupancy models (Chapter 19).

```
library(ubms)

# Load unmarked, format data and summarize
# Produces same as for unmarked in Section 19B.4 of course
library(unmarked)
summary( umf <- unmarkedFramePCount(y = C,
  siteCovs = data.frame(elev=mhbElevScaled, elev2=mhbElevScaled^2) ) )


unmarkedFrame Object

267 sites
Maximum number of observations per site: 3
Mean number of observations per site: 3
Sites with at least one detection: 186

Tabulation of y observations:
  0   1   2   3   4   5   6   7   8
322 185 120  71  39  39  11   8   6

Site-level covariates:
      elev             elev2
 Min.   :0.2193   Min.   :0.04809
 1st Qu.:0.5827   1st Qu.:0.33958
 Median :1.1165   Median :1.24649
 Mean   :1.1875   Mean   :1.82244
 3rd Qu.:1.8187   3rd Qu.:3.30749
 Max.   :2.7807   Max.   :7.73212
```

Of course, this is exactly the same data set, and summary, as we had for unmarked. To fit the model, the syntax is almost exactly the same as that in unmarked when using function pcount().

```
# Fit the model in ubms in parallel (ART 7 min)
options(mc.cores=4) # number of parallel cores to use
system.time(
   out19B.8 <- stan_pcount(~elev ~ elev + elev2, umf, chains=4))
out19B.8
ubms_est <- coef(out19B.8)     # Save estimates
```

Also the output looks almost exactly as the output from unmarked. Note that instead of the AIC, we are given the approximate loo-CV model selection criterion developed by Vehtari *et al.* (2017), as implemented in the loo package (see also Section 18.6).

```
Call:
stan_pcount(formula = ~elev ~ elev + elev2, data = umf, chains = 4)

Abundance (log-scale):
            Estimate    SD  2.5% 97.5% n_eff Rhat
(Intercept)    -3.37 0.288 -3.95 -2.83  1134 1.00
elev            8.85 0.565  7.74 10.00   954 1.00
elev2          -3.50 0.301 -4.09 -2.89   867 1.01

Detection (logit-scale):
            Estimate    SD  2.5% 97.5% n_eff Rhat
(Intercept)     2.17 0.344  1.52  2.88  1104    1
elev           -2.25 0.417 -3.14 -1.50  1106    1

LOOIC: 1788.418
Runtime: 5.944 min
```

We make our usual comparison of all the engines that we used so far in this chapter to fit the model to the simulated bullfinch data. We see that `ubms` is a bit off the remainder of the engines, but in fact is closer to the truth than the others. The differences are likely primarily due to `ubms` using by default a different set of priors than what we specified in the other engines. These priors can be changed but we'll leave this as an exercise for you.

```
# Compare estimates with truth
comp <- cbind(truth = truth, unmarked = unm_est, JAGS = jags_est,
  NIMBLE = nimble_est, Stan = stan_est, ubms = ubms_est)
print(comp, 3)
```

```
          truth unmarked  JAGS NIMBLE  Stan  ubms
alpha.lam  -3.0    -3.55 -3.50  -3.46 -3.48 -3.37
beta1.lam   8.5     9.20  9.13   9.06  9.06  8.85
beta2.lam  -3.5    -3.66 -3.63  -3.62 -3.57 -3.50
alpha.p     2.0     2.18  2.13   2.13  2.23  2.17
beta.p     -2.0    -2.25 -2.21  -2.18 -2.32 -2.25
```

We can obtain an estimate of the total population size across quadrats using the `posterior_predict` function provided by `ubms`, which yields posteriors of the latent abundance at each site.

```
# Compare total population size to truth
postN_ubms <- posterior_predict(out19B.8, "z") # Get posterior of latent
abundance
ubms_NtotalX <- apply(postN_ubms, 1, sum) # Calculate sumN for each draw
ubms_Ntotal <- median(ubms_NtotalX)
comp <- c(truth = true_Ntotal, naïve = naive_Ntotal,
  unmarked = unm_Ntotal, JAGS = jags_Ntotal, NIMBLE = nimble_Ntotal,
  Stan = stan_Ntotal, ubms = ubms_Ntotal)
print(comp, 2)
```

```
   truth    naïve unmarked     JAGS   NIMBLE     Stan     ubms
     908      534      979      965      981      998      973
```

19B.9 Do-it-yourself maximum likelihood estimates

Next, our glorious DIY-MLEs. This section will give us a look at what `unmarked` does under the hood when fitting this model using maximum likelihood ... or programs MARK (White & Burnham, 1999) or PRESENCE (Hines 2006) for that matter. The underlying likelihood math is the same as what we presented in section 19B.7 for Stan. Here, we're just translating the math into R code instead of Stan code.

We show two variants of the negative log-likelihood function (NLL). The first is faster, but the second is easier to understand. Our first NLL is from Panel 8.1. in Royle & Dorazio (2008). In addition to the arguments for the replicated count matrix (`C`) and the single site-covariate `x`, the R function defining the negative log-likelihood has arguments `K` and `nSites` and `nVisits`. The first is our setting for the upper integration bound *K* as previously discussed in the `unmarked` and Stan sections. The other two arguments are self-explanatory.

```
# Definition of NLL for Binomial N-mixture model for lizard counts
# Variant 1 (from Royle & Dorazio, 2008), faster than Variant 2 below
NLL <- function(param, C, x, K, nSites, nVisits){
  alpha.lam <- param[1]    # Abundance intercept (log scale)
  beta1.lam  <- param[2]   # Abundance slope on x
  beta2.lam  <- param[3]   # Abundance slope on x^2
  alpha.p   <- param[4]    # Detection intercept (logit scale)
  beta.p    <- param[5]    # Detection slope on x

  # Covariate model for expected abundance lambda ('GLM 1')
  lambda <- exp(alpha.lam + beta1.lam * x + beta2.lam * x^2)
  # Covariate model for detection probability p ('GLM 2')
  p <- plogis(alpha.p + beta.p*x)

  L <- rep(NA, 3) # Vector for likelihood contribution of each site
  tmp.like <- matrix(NA, nrow = K+1, ncol = nVisits)
  for(i in 1:nSites){
    gN <- dpois(0:K, lambda[i])
    gN <- gN / sum(gN)
    dvec <- C[i,]    # Extract counts at site i
    tmp.like[,1] <- dbinom(dvec[1], 0:K, p[i])
    tmp.like[,2] <- dbinom(dvec[2], 0:K, p[i])
    tmp.like[,3] <- dbinom(dvec[3], 0:K, p[i])
    likvec <- apply(tmp.like, 1, prod)
    L[i] <- sum(likvec * gN)  # Contribution to L from 1 site
  }
  NLL <- -1 * sum(log(L))
  return(NLL)
}
```

```
# Variant 2: much slower, but easier to understand than Variant 1

NLL <- function(param, C, x, K, nSites, nVisits){
  alpha.lam <- param[1]    # Abundance intercept (log scale)
  beta1.lam  <- param[2]   # Abundance slope on x
  beta2.lam  <- param[3]   # Abundance slope on x^2
  alpha.p   <- param[4]    # Detection intercept (logit scale)
  beta.p    <- param[5]    # Detection slope on x

  # Covariate model for expected abundance lambda ('GLM 1')
  lambda <- exp(alpha.lam + beta1.lam * x + beta2.lam * x^2)
  # Covariate model for detection probability p ('GLM 2')
  p <- plogis(alpha.p + beta.p*x)

  L <- rep(NA, nSites) # Vector for likelihood contribution of each site

  for(i in 1:nSites){
    pN <- dpois(0:K, lambda[i])
    pY <- rep(1, K+1)
    for (k in 0:K){
      for (j in 1:nVisits){
        pY[k+1] <- pY[k+1] * dbinom(C[i,j], k, p[i])
      }
    }
    L[i] <- sum(pN * pY)
  }
  NLL <- -1 * sum(log(L))
  return(NLL)
}
```

Using one or the other R functions for the NLL in the next block of code yields estimates in 23 or 55 seconds, respectively. Numerically, the solutions are of course identical.

```
# Minimize that NLL to find MLEs and get SEs (ART 23 or 55 sec)
inits <- c('alpha.lam' = 0, 'beta1.lam' = 0, 'beta2.lam' = 0,
  'alpha.p' = 0, 'beta.lp' = 0)
inits <- rep(0, 5)
names(inits) <- names(truth)
system.time( out19B.9 <- optim(inits, NLL, C = C, x = mhbElevScaled,
  K = max(C) + 100, nSites = nSites, nVisits = 3, hessian=TRUE,
  method = "BFGS", control=list(trace=1, REPORT=2)) )
get_MLE(out19B.9, 4)
diy_est <- out19B.9$par      # Save estimates


             MLE    ASE LCL.95 UCL.95
alpha.lam -3.554 0.3040 -4.150 -2.958
beta1.lam  9.205 0.5937  8.041 10.368
beta2.lam -3.661 0.3197 -4.287 -3.034
alpha.p    2.181 0.3523  1.491  2.871
beta.p    -2.252 0.4377 -3.110 -1.394

# Get AIC: 2 * deviance + 2 * number of parameters
(AIC <- 2 * out19B.9$value + 2 * length(out19B.9$par))


[1] 1788.085
```

We can compare these estimates with those obtained from `unmarked` as follows (not shown), and perhaps unsurprisingly, we find them to be identical.

```
# Compare with the solutions and AIC from unmarked
summary(out19B.4)                # not shown
```

That makes us pretty proud --- this is our first *N*-mixture model fit with likelihood inference 'by hand'. We can again use a parametric bootstrap to obtain estimates of the random effects (*N*) and of their sum across all 267 sample quadrats, as we did in Section 19B.4.2 and also in the Stan section above.

```
# Get estimate of sum N (ART 50 sec)
library(MASS)
Beta <- out19B.9$par
Sigma <- solve(out19B.9$hessian)
param_samples <- mvrnorm(1000, Beta, Sigma)
system.time(
  post_sumN <- get_post_sumN(param_samples, 30)
)
diy_Ntotal <- median(post_sumN)
```

The next block of code allows us to plot the empirical posterior distribution of the sum of the local abundance parameters (the *N*'s) for our DIY-MLEs of the *N*-mixture model fit to the Swiss bullfinches.

```
# Plot posterior of totalN from DIY-MLEs -- not shown
hist(post_sumN, main="Posterior of sumN from DIY MLEs", breaks = 40)
abline(v=median(post_sumN), col='blue', lwd = 5)
abline(v=true_Ntotal, col='red', lwd = 5)
legend("topright", lty=1, col=c('red','blue'),
legend=c('truth','estimate'), lwd = 4, cex = 1, bty = 'n')
```

19B.10 Likelihood analysis with TMB

In TMB we implement the model in the same way as we did in Stan and for our DIY-MLEs, and as it is implemented under the hood in both `unmarked` and `ubms`. That is, we define the marginal, or integrated, likelihood, wherein the latent states are summed out, and we are left with the parameters in the model for abundance. Again, we translate the math in section 19B.7, but this time into TMB code.

We use the same data set as we used for Stan. Note that here (as with Stan) we set the value of *K* at the maximum observed count plus 20, rather than adding 100 as does `unmarked` by default. We do this just to save time, since for our analysis here this value is sufficient.

```
# Bundle and summarize data
Kmin <- apply(C, 1, max)
K = max(C) + 20
dataList <- list(C=C, nSites=nSites, nVisits=nVisits,
  elev=mhbElevScaled, elev2= mhbElevScaled^2, Kmin=Kmin, K=K)
str(dataList)                # not shown
```

```
# Write TMB model file
cat(file="model19B_10.cpp",
"#include <TMB.hpp>

template<class Type>
Type objective_function<Type>::operator() ()
{
  //Describe input data
  DATA_MATRIX(C);
  DATA_INTEGER(nSites);
  DATA_INTEGER(nVisits);
  DATA_VECTOR(elev);
  DATA_VECTOR(elev2);
  DATA_INTEGER(K);
  DATA_IVECTOR(Kmin);

  //Describe parameters
  PARAMETER(alpha_lam);
  PARAMETER(beta1_lam);
  PARAMETER(beta2_lam);
  PARAMETER(alpha_p);
  PARAMETER(beta_p);

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

  vector<Type> lambda(nSites);
  vector<Type> p(nSites);
  vector<Type> lik(nSites);

  for (int i=0; i<nSites; i++){
    lambda(i) = exp(alpha_lam + beta1_lam * elev(i) + beta2_lam *
elev2(i));
    p(i) = invlogit(alpha_p + beta_p * elev(i));

    lik(i) = 0;
    for (int k=Kmin(i); k<(K+1); k++){
      Type pN = dpois(Type(k), lambda(i), false);
      Type pY = 1;
      for (int j=0; j<nVisits; j++){
        pY *= dbinom(C(i,j), Type(k), p(i), false);
      }
      lik(i) += pN * pY;
    }
    loglik += log(lik(i));
  }

  Type opt_elev = -beta1_lam / (2 * beta2_lam);
  ADREPORT(opt_elev);

  return -loglik;
}
")
```

```
# Compile and load TMB function
compile("model19B_10.cpp")
dyn.load(dynlib("model19B_10"))

# Provide dimensions and starting values for parameters
params <- list(alpha_lam = 0, beta1_lam = 0, beta2_lam = 0,
  alpha_p = 0, beta_p = 0)

# Create TMB object
out19B.10 <- MakeADFun(data = dataList, parameters = params,
                       DLL = "model19B_10", silent=TRUE)

# Optimize TMB object and print results
starts <- rep(0, 5)
opt <- optim(starts, fn = out19B.10$fn, gr = out19B.10$gr,
  method = "BFGS", hessian = TRUE)
(tsum <- tmb_summary(out19B.10))
tmb_est <- opt$par      # Save estimates
```

```
           Estimate Std. Error    LCL.95    UCL.95
alpha_lam -3.554198 0.30396203 -4.149952 -2.958443
beta1_lam  9.204768 0.59371395  8.041110 10.368426
beta2_lam -3.660544 0.31970568 -4.287155 -3.033932
alpha_p    2.181011 0.35227553  1.490564  2.871459
beta_p    -2.252471 0.43773936 -3.110424 -1.394517
opt_elev   1.257295 0.05350854  1.152420  1.362170
```

To get the number of simulated bullfinches in the 267 sample quadrats, we can parametrically bootstrap the same function that we had used for Stan and for our DIY-MLEs above. This is the same application of a parametric bootstrap without refitting the model as we had encountered in Section 19B.4.2. Also note that the number of bootstrap samples chosen in the next code block will define the smoothness of the posterior distribution of the estimated quantity. Here we will choose 10,000 to obtain a really nice picture. We use the median yet again as a point estimator.

```
# Get estimate of sum N from TMB solutions and then plot
bootstrap.n <- 10000       # Choose how many samples from MVN to draw
Beta <- tmb_est
Sigma <- solve(opt$hessian)
param_samples <- mvrnorm(bootstrap.n, Beta, Sigma)
system.time(
  post_sumN <- get_post_sumN(param_samples, 30)  # ART 9 min
)
tmb_Ntotal <- median(post_sumN)

# Plot (not shown)
hist(post_sumN, main="Posterior of sum(N) from TMB", breaks=50)
abline(v=true_Ntotal, col='red', lwd = 5)
abline(v=post_sumN, col='blue', lwd = 5)
legend('topright', col=c('red','blue'), lty=1, lwd = 4,
legend=c('truth','estimate (median)'), bty = 'n')
```

19B.11 Comparison of the estimates

Finally, our official grand comparison of the regression parameters in both submodels of the binomial *N*-mixture model. Remember that we have solutions based on maximum likelihood (`unmarked`, DIY and TMB) and on Bayesian posterior inference (JAGS, NIMBLE, Stan and `ubms`).

```
# Compare results with truth
comp <- cbind(cbind(truth = truth, unmarked = unm_est, JAGS = jags_est,
   NIMBLE = nimble_est, Stan = stan_est, ubms = ubms_est,
   DIY = diy_est, TMB = tmb_est))
print(comp, 4)
```

```
          truth unmarked   JAGS NIMBLE   Stan   ubms    DIY    TMB
alpha.lam  -3.0   -3.554 -3.498 -3.458 -3.483 -3.375 -3.554 -3.554
beta1.lam   8.5    9.205  9.129  9.056  9.061  8.852  9.205  9.205
beta2.lam  -3.5   -3.661 -3.635 -3.622 -3.574 -3.500 -3.661 -3.661
alpha.p     2.0    2.181  2.127  2.127  2.227  2.166  2.181  2.181
beta.p     -2.0   -2.252 -2.212 -2.180 -2.324 -2.246 -2.252 -2.252
```

… and the comparison for the total population size within the 267 surveyed quadrats.

```
# Compare total population size to truth
comp <- c(truth = true_Ntotal, naive = naive_Ntotal,
  unmarked = unm_Ntotal, JAGS = jags_Ntotal, NIMBLE = nimble_Ntotal,
  Stan = stan_Ntotal, ubms = ubms_Ntotal, DIY = diy_Ntotal,
  TMB = tmb_Ntotal)
print(comp, 2)
```

```
   truth    naive unmarked     JAGS   NIMBLE     Stan
     908      534      979      965      981      998

    ubms      DIY      TMB
     973      975      978
```

Overall, we find what we have almost always found in this comparison throughout the book: for practical purposes, most estimates are numerically identical.

19B.12 Summary and outlook

Chapter 19 in the ASM book and this chapter 19B feature two special types of species distribution models (SDMs), which include an explicit submodel for the measurement error in presence/absence and abundance data, respectively. After the occupancy model for detection/nondetection data of Chapter 19, we have now presented the other foundational hierarchical model for distribution and abundance: the binomial *N*-mixture model of Royle (2004a) for count data. We have showcased one of the simplest possible examples of this model: a static, or "single-season", model that assumes closed populations at every surveyed site over the duration of all repeated surveys. The closure assumption combined with the repeated-measures design enables one to separately estimate the state model for latent abundance *N* and the observation model, which is governed by the detection probability parameter *p*.

We have illustrated one of the main strengths of this type of hierarchical model: that we can separately estimate effects of a single covariate on both the state and on the observation

model (Kéry 2008). In our simulated example with bullfinch counts from the Swiss monitoring of common breeding birds (MHB; Schmid *et al.*, 2004), site elevation affected both abundance and detection. A traditional species distribution model is unable to tease apart the two processes that underlie the observed counts; it underestimated abundance and yielded a biased assessment of the elevation at which bullfinch abundance is greatest. In contrast, the *N*-mixture model led to unbiased estimates of abundance and of the optimal elevation, at least in our best-case scenario, where the data-generation and the data-analysis models matched exactly.

Comparison of the estimates among our model-fitting engines has shown the typical numerical similarity that we have come to expect by now. Admittedly, this may not always be so, e.g., when sample sizes are small, priors are informative, or with skewed posterior distributions. However, we think that this is very comforting and provides support for our belief that ecologists should be eclectic in their choice of inference framework for a statistical model and should understand both likelihood and Bayesian methods.

As with the occupancy model in Chapter 19, we have used functions in two specialized species distribution modeling packages: `unmarked` (Fiske & Chandler, 2011; Kellner *et al.*, 2023) and `ubms` (Kellner *et al.*, 2022). Both contain fitting functions and much additional functionality for a wide range of SDMs which accommodate and therefore correct for false-negative, and in some cases false-positive, measurement errors. While `unmarked` uses maximum likelihood, `ubms` is a wrapper for Stan and thus uses Bayesian posterior inference. Both enable the user to specify random effects (e.g., for regions or for observers), but the treatment of random effects is arguably smoother with Bayesian inference than with maximum likelihood. For example, `ubms` also contains spatial modeling functions by use of restricted spatial regression, or RSR, a similar model for residual spatial autocorrelation as the widely used conditionally autoregressive (CAR) model (Johnson *et al.*, 2013). A wide range of occupancy and *N*-mixture models can also be fit using the new R packages `spOccupancy` and `spAbundance` (Doser *et al.*, 2022, 2024), which provide Bayesian posterior inference for models for single or multiple species, with or without species interactions, and for spatial or spatiotemporal settings, among others.

We have illustrated two further important topics in this chapter. One is spatial prediction, and the other is uncertainty assessment for functions of parameters, or derived quantities, using two frequentist methods: the delta method and the parametric bootstrap.

Prediction means to compute the expected values of the response, or more generally of a parameter, as a function of some covariate in the model, as we vary the value of that covariate. Prediction is an important part of modeling and one that we have illustrated throughout the book, e.g., in Chapters 5, 7, 8, 9, and 13. Making predictions from a statistical model serves two important goals: presenting the results of an analysis, typically in a figure, and understanding what the model is telling us in the first place. In very simple models, we can often understand the message of the model by simply inspecting the parameter estimates and their sign. For more complex models, e.g., with interactions, non-linearity or other complications, this may become hard or impossible. In those cases, making predictions and plotting them against the covariates may be our main way to understand what the model is telling us.

In this chapter, we have made extensive use of spatial predictions, which we can get from any regression model with spatially indexed covariates. Notably, any such model can be used as a species distribution model by simply projecting the estimated regression relationships into geographic space, using for prediction at each site (or quadrat, pixel) the actual values of the covariates there (Royle *et al.*, 2005). In our case, working with simulated data for Swiss bullfinches, we have produced abundance-based species distribution maps which depict the expected abundance of the species at the 1 km^2 scale for the entire country. Importantly, we have also produced maps of the associated estimation uncertainty, e.g., in the form of standard errors or posterior standard deviations of these predictions, or by the lower and upper limits of

confidence or credible intervals. Finally, we have seen how, under an assumption of independence of the sites, it is trivially easy to obtain national population size estimates by simply adding up the predicted values over all sites. Obviously, regional totals could be obtained just as easily, by adding up the predictions for sites not nationwide, but only over specific regions of a country.

Bayesians use a single "method", the posterior distribution, to produce both point estimates and uncertainty assessments for all parameters in a model. We have seen in Chapter 2 that when using maximum likelihood, we are not so lucky, but instead use the likelihood function for parameter estimation, and then a variety of methods exist to come up with uncertainty assessments for these estimates. We have illustrated two of these methods for the aim of obtaining the standard error of a function of the basic parameters in the model: the delta method and a simple parametric bootstrap (see also Section 2.5). These are widely used in frequentist software such as `unmarked`, which uses the delta method for variance estimation of predictions in many model fitting functions, while some newer functions use the parametric bootstrap instead, which is based on the assumed multivariate normality of the estimates. Seeing both methods in action will arguably help you get more comfortable with software that uses this technology under the hood.

Returning to the binomial mixture model, we make an important observation about the abundance parameter. The same observation applies in an analogous way for the occupancy parameter in the occupancy model in Chapter 19 of the book. Even when correcting for imperfect detection, the interpretation of the abundance parameter N_i may not necessarily be what we might want it to be: the number of individuals that permanently reside within a defined plot of land. The reason is that animals may move around, so the effective sampling area will often be greater than the nominal sampling area. Hence, the estimate of N_i will often refer to a larger area and we don't exactly know the size of it. The magnitude of the discrepancy between the nominal and the effective sampling area depends on two things: the typical scale of the movement of the study species and the time frame of the repeated surveys. The discrepancy will be greater for greater dispersal and a longer total survey period. The estimate of N_i then refers to the number of animals that ever use the area over the entire during of the sampling. The same reasoning goes for the interpretation of the occupancy parameter in occupancy models in the face of temporary emigration (MacKenzie *et al.* 2018).

If we want to circumvent this difficulty, other sampling protocols and associated modeling frameworks must be used, such as *distance sampling* (Buckland *et al.* 2001, 2015) or *spatial capture-recapture* methods (Efford 2004; Borchers & Efford, 2008; Royle & Dorazio 2008; Royle & Young 2008; Efford *et al.*, 2009; Royle *et al.*, 2014). We note also that adding up spatial predictions of the expected abundance to obtain estimates of total population size, as we have done multiple times in this chapter, is only valid under the assumption of independence of sites. Thus, in so doing we assume that the effective sampling area associated with a site is not greater than 1 km^2 and does not overlap with that of its neighbors (see Section 6.10 in Kéry & Royle, 2016).

This issue is not unique to the binomial mixture (or the occupancy) model; rather, it is important for most models that ignore imperfect detection or other kinds of species distribution measurement errors. Thus, a binomial mixture model solves the problem of imperfect detection when interpreting (i.e., analyzing) count data. But the issue of how exactly abundance should be interpreted (e.g., whether as a density for a known area) may remain a challenge.

Binomial mixture models offer great opportunities for improved estimation of animal or plant abundance, by correcting for imperfect detection probability (p). Essentially, the model is simply a generalized version of a Poisson regression model that accommodates imperfect

detection; when $p = 1$, we are back to a classical Poisson generalized linear model (see Chapters 11–13). However, it is far more complex than a simpler Poisson GLM and also more complex than a conventional hierarchical Poisson GLMM (Chapter 14) and fitting it in practice may be challenging sometimes (Goldstein & de Valpine, 2022).

There are many extensions to the basic *N*-mixture model that we have shown here. These include different observation protocols (e.g., Wyatt 2002; Royle 2004b; Royle *et al.*, 2004; Strebel *et al.*, 2021), multiple species (Yamaura *et al.*, 2012), open populations (Dail & Madsen 2011), and multiple classes of individuals, such as young and adult (Zipkin *et al.*, 2014). Most are synthesized in the two *Applied Hierarchical Modeling*, or AHM, books; see Kéry & Royle (2016, 2021).

References

Andrewartha, H.G. & Birch, L.C. 1954. *The distribution and abundance of animals*. University of Chicago Press, Chicago.

Barker, R. J., Schofield, M. R., Link, W. A., & Sauer, J. R. (2018). On the reliability of *N*-mixture models for count data. Biometrics, 74, 369–377.

Berger, J.O., Liseo, B., & Wolpert, R.K. 1999. Integrated likelihood methods for eliminating nuisance parameters. *Statistical Science*, 14, 1–22.

Bibby, C.J., Burgess, N.D., Hill, D.A. & Mustoe, S. 2000. *Bird Census Techniques*, Second Edition, Academic Press.

Borchers, D.L. & Efford, M.G. 2008. Spatially explicit maximum likelihood methods for capture-recapture studies. *Biometrics*, 64, 377–385.

Buckland, S.T., Anderson, D.R., Burnham, K.P., Laake, J.L., Borchers, D.L. & Thomas, L. 2001. *Introduction to distance sampling.* Oxford University Press, Oxford.

Buckland, S.T., Rexstad, E.A., Marques, T.A., & Oedekoven, C.S. 2015. *Distance sampling: methods and applications*. Springer, Cham, Switzerland.

Couturier, T., Cheylan, M., Bertolero, A., Astruc, G., & Besnard, A. 2013. Estimating abundance and population trends when detection is low and highly variable: A comparison of three methods for the Hermann's tortoise. *Journal of Wildlife Management*, 77, 454–462.

Dennis, E.B., Morgan, B.J., Ridout, M.S. (2015). Computational aspects of *N*-mixture models. *Biometrics*, 71, 237–246.

Dail, D. & Madsen, L. 2011. Models for estimating abundance from repeated counts of an open population. *Biometrics*, 67, 577–587.

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

Dorazio, R.M., 2007. On the choice of statistical models for estimating occurrence and extinction from animal surveys. *Ecology*, 88, 2773–2782.

Doser, J.W., Finley, A.O., Kéry, M., Zipkin, E.F. 2022. `spOccupancy`: An R package for single species, multispecies, and integrated occupancy models. *Methods in Ecology and Evolution*, 13, 1670–1678.

Doser, J.W., Finley, A.O., Kéry, M., Zipkin, E.F. 2024. `spAbundance`: An R package for single-species and multi-species spatially-explicit abundance models. *Methods in Ecology and Evolution*, 15, 1024–1033.

Duarte, A., Adams, M. J., & Peterson, J. T. 2018. Fitting *N*-mixture models to count data with unmodeled heterogeneity: Bias, diagnostics, and alternative approaches. *Ecological Modelling*, 374, 51–59.

Efford, M. 2004. Density estimation in live-trapping studies. *Oikos*, 106, 598–610.
Efford, M.G., Borchers, D.L. & Byrom, A.E. 2009. Density estimation by spatially explicit capture-recapture: likelihood-based methods. pp. 255–269 in D.L. Thomson, E.G. Cooch, M.J. Conroy (eds.) *Modeling demographic processes in marked populations*. Springer, New York.
Fiske, I. & Chandler, R. 2011. `unmarked`: an R package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software*, 43, 1–23.
Goldstein, B.R., & de Valpine, P. 2022. Comparing *N*-mixture models and GLMMs for relative abundance estimation in a citizen science dataset. *Scientific Reports*, 12, 12276.
Hines, J.E. 2006. *PRESENCE 3.1 Software to estimate patch occupancy and related parameters*. http://www. mbr-pwrc. usgs.gov/software/presence.html.
Johnson, D.S., Conn, P.B., Hooten, M.B., Ray, J.C. & Pond, B.A. 2013. Spatial occupancy models for large data sets. *Ecology*, 94, 801–808.
Joseph, L.N., Elkin, C., Martin, T.G. & Possingham, H. 2009. Modeling abundance using N-mixture models: the importance of considering ecological mechanisms. *Ecological Applications*, 19, 631–642.
Joseph, M.B., 2020b. A step-by-step guide to marginalizing over discrete parameters for ecologists using Stan. Accessed on 16 November 2023. https://mbjoseph.github.io/posts/2020-04-28-a-step-by-step-guide-to-marginalizing-over-discrete-parameters-for-ecologists-using-stan/.
Kellner, K.F., N.L. Fowler, T.R. Petroelje, T.M. Kautz, D.E. Beyer Jr., J.L. Belant. 2022 A. `ubms`: An R package for fitting hierarchical occupancy and *N*-mixture abundance models in a Bayesian framework. *Methods in Ecology and Evolution*, 13, 577–584.
Kellner, K.F., Smith, A.D., Royle, J.A., Kéry, M., Belant, J.L., Chandler, R.B. 2023. The `unmarked` R package: Twelve years of advances in occurrence and abundance modeling in ecology. *Methods in Ecology and Evolution*, 14, 1408–1415.
Kéry, M. 2010. *Introduction to WinBUGS for Ecologists. - A Bayesian approach to regression, ANOVA, mixed models and related analyses.* Academic Press, Burlington.
Kéry, M., 2008. Estimating abundance from bird counts: binomial mixture models uncover complex covariate relationships. *The Auk*, 125, 336–345.
Kéry, M. 2018. Identifiability in N-mixture models: A large-scale screening test with bird data. *Ecology*, 99, 281–288.
Kéry, M., Royle, J.A. & Schmid, H. 2005. Modeling avian abundance from replicated counts using binomial mixture models. *Ecological Applications*, 15, 1450–1461.
Kéry, M. & Royle, J.A. 2010. Hierarchical modeling and estimation of abundance in metapopulation designs. Journal of Animal Ecology, 79, 453–461.
Kéry, M. & Royle, J.A. 2016. *Applied hierarchical modeling in ecology—Modeling distribution, abundance and species richness using R and BUGS.* Volume 1: Prelude and Static Models. Elsevier / Academic Press.
Kéry, M. & Royle, J.A. 2021. *Applied hierarchical modeling in ecology—Modeling distribution, abundance and species richness using R and BUGS.* Volume 2: Dynamic and Advanced Models. Elsevier / Academic Press.
Kéry, M. & Schmidt, B.R. 2008. Imperfect detection and its consequences for monitoring for conservation. *Community Ecology*, 9, 207–216.
Kéry, M., Royle, J.A., Plattner, M. & Dorazio, R.M. 2009. Species richness and occupancy estimation in communities subject to temporary emigration. *Ecology*, 90, 1279–1290.
Knape, J., Arlt, D., Barraquand, F., Berg, A., Chevalier, M., Pärt, T., Ruete, A., Żmihorski, M. 2018. Sensitivity of binomial N-mixture models to overdispersion: The importance of assessing model fit. *Methods in Ecology and Evolution*, 9, 2102–2114.
Knaus, P., Antoniazza, S., Wechsler, S., Guélat, J., Kéry, M., Strebel, N. & Sattler, T. 2018. *Brutvogelatlas 2013–2016. Bestandsentwicklung der Brutvögel der Schweiz und des*

Fürstentums Liechtensteins (Swiss Breeding Bird Atlas 2013–2016). Schweizerische Vogelwarte, Sempach.
Krebs, C.J. 2009. *Ecology: The experimental analysis of distribution and abundance.* 6 Ed. Benjamin Cummings, San Francisco.
Link, W.A. & Sauer, J.R. 2002. A hierarchical analysis of population change with application to Cerulean warblers. *Ecology*, 83, 2832–2840.
Link, W.A., Schofield, M.R., Barker, R.J., & Sauer, J.R. 2018. On the robustness of *N*-mixture models. *Ecology*, 99, 1547–1551.
MacKenzie, D.I., Nichols, J.D., Royle, J.A., Pollock, K.H., Hines, J.E. & Bailey, L.L. 2017. *Occupancy estimation and modeling: inferring patterns and dynamics of species occurrence.* Second Edition. Elsevier, San Diego.
Madsen, L. & Royle, J.A. 2023. A review of *N*-mixture models. *WIREs Computational Statistics*, 15, e1625.
Ponisio, L.C., de Valpine, P., Michaud, N., & Turek, D. 2020. One size does not fit all: Customizing MCMC methods for hierarchical models using NIMBLE. *Ecology & Evolution*, 10, 2385–2416.
Powell, L. A. 2007. Approximating variance of demographic parameters using the delta method: a reference for avian biologists. *Condor*, 109, 949–954.
Royle, J.A. 2004a. *N*-mixture models for estimating population size from spatially replicated counts. *Biometrics*, 60, 108–115.
Royle, J.A. 2004b. Generalized estimators of avian abundance from count survey data. *Animal Biodiversity and Conservation*, 27.1, 375–386.
Royle, J.A. & Dorazio, R.M. 2006. Hierarchical models of animal abundance and occurrence. *Journal of Agricultural, Biological and Environmental Statistics*, 11, 249–263.
Royle, J.A. & Dorazio, R.M. 2008. *Hierarchical modeling and inference in ecology. The analysis of data from populations, metapopulations and communities.* Academic Press, New York.
Royle, J.A. & Young, K.G. 2008. A hierarchical model for spatial capture-recapture data. *Ecology*, 89, 2281–2289.
Royle, J.A., Dawson, D.K. & Bates, S. 2004. Modeling abundance effects in distance sampling. *Ecology*, 85, 1591–1597.
Royle, J.A., Nichols, J.D. & Kéry, M. 2005. Modelling occurrence and abundance of species when detection is imperfect. *Oikos*, 110, 353–359.
Royle, J.A., Chandler, R.B., Sollmann, R. & Gardner, B. 2014. *Spatial Capture-Recapture.* Academic Press.
Schmid, H., Zbinden, N. & Keller, V. 2004. *Überwachung der Bestandsentwicklung häufiger Brutvögel in der Schweiz* (Surveillance monitoring of common breeding birds in Switzerland). Report, Schweizerische Vogelwarte, Sempach.
Strebel, N., R. Dröschmeister, H. Schmid, I. Stützle, S. Trautmann & J. Wahl. 2020. Comparing territory-based and individual-based population trend estimates in the monitoring of common breeding birds. *Vogelwelt*, 140, 183–206.
Strebel, N., Fiss, C.J., Kellner, K.F., Larkin, J.L., Kéry, M. & Cohen, J. 2021. Estimating abundance based on time-to-detection data. *Methods in Ecology and Evolution*, 12, 909–920.
Vehtari, A., Gelman, A., Gabry, J. 2017. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computation*, 27, 1413–1432.
ver Hoef, J.M. & Jansen, J.K. 2007. Space-time zero-inflated count models of harbour seals. *Environmetrics*, 18, 697–712.
Wenger, S.J. & Freeman, M.C. 2008. Estimating species occurrence, abundance, and detection probability using zero-inflated distributions. *Ecology*, 89, 2953–2959.
White, G.C. & Burnham, K.P. 1999. Program MARK: survival estimation from populations of marked animals. *Bird Study*, 46, 120–139.

Williams, B.K., Nichols, J.D. & Conroy, M.J. 2002. *Analysis and management of animal populations.* Academic Press, San Diego.

Wyatt, R.J., 2002. Estimating riverine fish population size from single-and multiple-pass removal sampling using a hierarchical model. *Canadian Journal of Fisheries and Aquatic Sciences*, 59, 695–706.

Yamaura, Y., Royle, J.A., Shimada, N., Asanuma, S., Sato, T., Taki, H. & Makino, S. 2012. Biodiversity of man-made open habitats in an underused country: a class of multispecies abundance models for count data. *Biodiversity and Conservation*, 21, 1365–1380.

Yackulic, C.B., Dodrill, M., Dzul, M., Sanderlin, J.S., Reid, J.A., 2020. A need for speed in Bayesian population models: a practical guide to marginalizing and recovering discrete latent states. *Ecological Applications*, 30, e02112.

Zipkin, E.F., Thorson, J.T., See, K., Lynch, H.J., Grant, E.H.C., Kanno, Y., Chandler, R.B., Letcher, B.H. & Royle, J.A. 2014. Modeling structured population dynamics using data from unmarked individuals. *Ecology*, 95, 22–29.