CHAPTER

INTRODUCTION TO DATA SIMULATION

4

CHAPTER OUTLINE

4.1 WHAT DO WE MEAN BY DATA SIMULATION, AND WHY IS IT SO TREMENDOUSLY USEFUL?

By data simulation, we simply mean the generation of random numbers from a stochastic process that is described by a series of distributional statements, such as $\alpha_i \sim Normal(\mu, \sigma^2_\alpha)$ and $y_{ij} \sim Normal(\alpha_i, \sigma^2)$, for a normal-normal mixed model; see Section 3.4.1. Data simulation is so exceedingly useful for your work as a quantitative ecologist, and moreover is done so frequently in this book, that we dedicate a whole chapter to it. Here is why:

1. *Truth is known*: We know the values of simulated parameters, and we can therefore compare the model estimates with those. Fitting a model and obtaining parameter estimates that resemble the input values is a good check that your coding in BUGS (or any other language) is right, and that the algorithmic MCMC black box in BUGS is doing the right thing.
2. *Calibrate derived model parameters*: Sometimes the effect of one or more parameters on the desired output of a model is unclear, such as the values of apparent survival and recruitment on the extinction probability of a population. Being able to tune the parameters such that meaningful data sets arise can be useful. Related to that, simulations can be regarded as controlled experiments, as simplified versions of a real system, in order to test how varying certain parameters affects estimates of other parameters—e.g., for a sensitivity analysis (T. L. Crewe, *pers. comm.*). Controlled experiments are impossible in many ecological studies, and simulation is one way to approximate them.
3. *Sampling error*: This can be described as the natural variability of the data and of the statistics computed from them, such as sample means or other parameter estimates (we find "error" a slightly

Applied Hierarchical Modeling in Ecology. http://dx.doi.org/10.1016/B978-0-12-801378-6.00004-7

misleading term and think that "natural variability" or even noise would be better terms). This variability arises because we have not measured every member in a heterogeneous statistical population of interest. The magnitude of sampling error determines our measures of statistical precision (e.g., standard errors and confidence intervals), which can be computed based on theory even from a single sample (i.e., from a single data set). In our experience, most ecologists find it very challenging to grasp the concept of sampling error—i.e., the natural variability among a hypothetical large collection of replicate data sets—when all they have is usually just a single replicate. Repeatedly turning the handle on the data-generating stochastic process represented by your data simulation code, and observing the resulting variability among simulated data sets or things computed from them such as parameter estimates, is a fantastic learning experience because it allows you to actually *observe* sampling error directly.

4. *Check the frequentist operating characteristics of estimators (e.g., bias, precision)*: It can be very useful to see how good your estimates are expected to be for given sample sizes and parameter values. To assess the estimator bias (or "is my estimate on target on average?") or precision (or "how repeatable, or variable around their averages, are the estimates?"), it is most straightforward to analyze a large number of simulated data sets for different choices of sample size or parameter values, and compute the difference between the mean of the estimates and truth (bias), and the variance of the estimates (precision).
5. *Power analyses*: Power is the probability to detect an effect in the data when it is really there. Analyzing a large number of simulated data sets is the most flexible way to estimate the power of a sampling design and associated analysis method. A closely related problem is the determination of the necessary sample size to detect an effect of a certain magnitude with a chosen probability (power).
6. *Check the identifiability/estimability of model parameters*: The mere fact that we can fit a certain model to a certain data set and get some numbers out does not guarantee that we actually have the "right" kind of data to inform every parameter. A parameter may be intrinsically or extrinsically unidentifiable, the latter refers to the case where there is no information in our data set to obtain an estimate of it. Intrinsic unidentifiability refers to the case where the structure of a model does not allow us in principle to estimate a certain parameter from a certain type of data set. A famous example in ecological statistics is in the Cormack-Jolly-Seber model with full time-dependence, where the last survival and recapture probabilities are confounded, and only their product can be estimated (Kéry and Schaub, 2012, p. 217–220). With extrinsic unidentifiability, we cannot estimate a certain parameter because of the vagaries of a particular data set. For example, in an interaction-effects ANOVA model, we will not be able to estimate all interaction terms if some combinations of the factor levels are not observed. Identifiability is always a worry with complex models (Cressie et al., 2009; Lele, 2010), and it is compounded in Bayesian analyses where we will always obtain an estimate: when there is no information in the data set to inform the parameter estimate, the estimate will simply be determined by the prior (see Section 5.5.2 for a striking example). Hence, in a strictly technical sense, all parameters are always identifiable in a Bayesian analysis when proper prior distributions are used. Practically, however, we have exactly the same problem as we would have in a frequentist analysis, because a parameter estimate entirely determined by the prior will likely be unsatisfactory to most. To make things more complicated, there are intermediates between estimability and nonestimability (Catchpole et al., 2001). For simple models or for certain classes of models, a lot is known about the parameters that are intrinsically identifiable (Cole, 2012), and for some, relatively well-worked-out methods exist to check parameter identifiability (Catchpole and Morgan, 1997; Catchpole et al., 2002; Choquet and Cole, 2012; Cole, 2012). This is not the case for the vast majority of hierarchical models, however, and extrinsic unidentifiability can never be known beforehand. Hence, perhaps the most

straightforward approach to check estimability in practice is to generate many replicate data sets under a model for various sets of parameter values, estimate the parameters, and see whether the estimates cluster around the data-generating values as they should if they are estimable.

7. *Check for the robustness of estimators and effects of assumption violations*: An assumption violation can be loosely defined as the presence of an "important" effect in the data-generating model and its absence in the data-analysis model. "Important" means that this absence has a noticeable effect on the quality of the desired inference. A straightforward way to gauge the robustness of our model to the violation of an assumption such as "Y does not vary among individuals" is to: generate data under the more general model where Y *does* vary among individuals, and then analyze the data with the restricted model where Y does *not* vary among individuals. Repeating this a large number of times (e.g., 100–1000) will allow us to say how influential the violation of this assumption is for our inference, e.g. by introducing bias in the other estimates. Often, as few as 5–10 simulation replicates may be enough to give you a pretty good idea of the broad patterns.
8. Finally, *data simulation provides proof that you understand a model*: If you can simulate data under a certain model, then it is likely that you really understand that model. Simulating data sets is the opposite of analyzing a data set: you assemble a data set in the data simulation procedure and then break it down again using the analysis procedure. Analyzing data is like fixing a broken motorbike: to be able to fix the bike, you must really know all its parts and how they relate to each other. If you can take apart a motorbike and then reassemble the parts into a functioning bike, you prove to yourself that you understand how the bike works, and you will be better able to diagnose and fix problems when they arise. In a similar vein, if you can assemble a data set using its "ingredients" (parameters, covariates), you will certainly understand what the parameter estimates mean that come out of the analysis. In the same spirit, we often use data simulation in this book as a means of explaining a model or a process. We believe that this can be a very important type of explanation that is complementary to equations and may actually be much more easily understood by many ecologists than equations.

In the next section, we will use R to walk you through the generation of a data set in great detail. We build a typical point count data set, where we have one count from each of a number of sites for each of a number of surveys that are conducted over a short period such as a breeding season. After that, we package the essential parts of that R code into a function. Functions make it much easier to repeatedly execute the simulation code, and allow you to easily change key settings such as sample size or parameter values without altering the underlying code.

4.2 GENERATION OF A TYPICAL POINT COUNT DATA SET

The data set we assemble is perhaps the canonical example of a count data set in nature as we see it, and for which we employ hierarchical models to learn about the features of the two processes that have generated them: the ecological and observation models. Hence, in this chapter we also illustrate, using data simulation, the way in which observed counts of some imaginary species arise as a result of the actions of an ecological process governing spatiotemporal variation in abundance, and of an observation process embodied by two possible forms of binary measurement error. We also emphasize that occurrence is simply an information-reduced summary of abundance, something we have seen before (e.g., in Section 3.3.6). In essence, we generate a prototypical data set that embodies our view of how spatially replicated observations of the biological abundance state typically arise.

To be more concrete, we give a name to our imaginary species and call it a great tit (*Parus major*; Figure 4.1), a small passerine widespread in Eurasia. We generate a data set that contains J replicate

FIGURE 4.1

The famous great tit *(Parus major; J. Peltomäki)*.

counts at each of M sites under the "closure" assumption: that the counts at a site take place in such a short time that abundance N at the site does not change. See Section 6.9 for a typical breeding bird survey that produces data of this kind and for an analysis of a real data set on great tits.

You will recognize the same concepts in data simulation that we will later apply in model building for making inferences about distribution and abundance: we clearly distinguish the ecological process that generates a (partially) latent state of abundance, N, from the observation process that produces the observed data, the counts of great tits. We assume that the observation process is governed by imperfect detection only, and that false-positives are absent. To make the example a little more realistic, we further build into abundance the effects of elevation and forest cover, plus their interaction, by assuming that abundance decreases linearly with elevation and increases linearly with forest cover, and that the two covariates interact negatively. We introduce these abundance effects on the log scale, as is customary in Poisson GLMs (see Chapter 3).

In our simulation, we make explicit that not every great tit in the area sampled at each site will be detected; i.e., we are faced with the kind of binary measurement error described in Chapter 1. Some tits may not sing during our survey, or they may sing but we are distracted by another tit or another vocalizing species or … Indeed, there are a myriad of reasons for why we may fail to detect an individual bird in nature. Thus, we will only count an individual tit with a detection probability of p, which for the sake of illustration we will make dependent (on the logit scale) on one site covariate, elevation,

and on one sampling covariate, wind speed. Birds sing less and are more difficult to hear in windier conditions. We assume that detection probability is positively related to elevation and negatively related to wind speed. Note that one covariate, elevation, affects both the ecological state (abundance) and its observation, or measurement (detection probability). We do this on purpose, because this is quite likely to happen in nature sometimes. The hierarchical models in this book are well able to tease apart such complex relationships (e.g., Kéry, 2008; and Chapters 12–13 in Kéry and Schaub, 2012). Finally, we could add an interaction effect between elevation and wind speed in detection, but we will not. In the R code, we allow you to build in such an interaction effect so that you can experiment with it if you like, but for now we drop this interaction effect by setting it to zero. In summary, we generate data under the following model, where sites are indexed i and replicate counts j, and α_3 is set to zero:

Ecological model:

$$N_i \sim Poisson(\lambda_i)$$

$$\log(\lambda_i) = \beta_0 + \beta_1 * elev_i + \beta_2 * forest_i + \beta_3 * elev_i * forest_i$$

Observation model:

$$C_{ij} \sim Binomial(N_i, p_{ij})$$

$$\text{logit}(p_{ij}) = \alpha_0 + \alpha_1 * elev_i + \alpha_2 * wind_{ij} + \alpha_3 * elev_i * wind_{ij}$$

We will simulate the data from "the inside out" and from top to bottom. First, we choose the sample size and create values for the covariates. Second, we choose parameter values for the ecological model, assemble the expected abundance λ, and draw the Poisson random variables N. Third, we choose parameter values for the observation model, assemble detection probability p, and draw the binomial random variables C (i.e., the observed counts of great tits).

4.2.1 INITIAL STEPS: SAMPLE SIZE AND COVARIATE VALUES

We first choose the sample size; number of sites, and replicate counts at each site.

```
M <- 267   # Number of spatial replicates (sites)
J <- 3     # Number of temporal replicates (counts)
```

Next, we create values for the covariates. We have elevation and forest cover as *site covariates*: they differ by site only, not by survey. And we have wind speed, which is a *sampling* or *observational covariate*; it varies potentially by both site and replicate. We will simply fill arrays with zero-mean random numbers for these covariates, so that all three covariates are centered on zero and do not extend too far on either side of zero. In real data analyses, we typically center or scale our covariates to avoid numerical problems with finding the MLEs and getting convergence of the Markov chains. We here ignore one feature of real life, which is that the covariates are typically not independent of each other (e.g., forest cover may be related with elevation), but this is not relevant for us now.

To initialize the random number generator used to obtain our data, we can add the following line at the start of the simulation:

```
set.seed(24)   # Can choose seed of your choice
```

We then always obtain the same data set and therefore, up to Monte Carlo error in the MCMC analysis, also get the same estimates. For reasons explained here (http://www.mbr-pwrc.usgs.gov/pubanalysis/kerybook/), we often prefer not to set a seed for the random number generators when

simulating a data set: basically, we feel that the actual data set at hand is fairly unimportant in most analyses. Your real interest usually lies in the stochastic process that has generated your data set simply as one *possible* realization, and this process is best studied by replicate data sets. By not singling out one particular data set as the "right" one, we emphasize the importance of the process that is behind the actual data set. Further, repeatedly running the code without a seed will give you a feel for sampling error. Still, for presentation purposes we use a seed here and generate values for elevation, forest cover, and wind speed that are all scaled to a range of (−1,1).

```
elev <- runif(n = M, -1, 1)                          # Scaled elevation of a site
forest <- runif(n = M, -1, 1)                        # Scaled forest cover at each site
wind <- array(runif(n = M*J, -1, 1), dim = c(M, J))  # Scaled wind speed
```

We now have covariate vectors of the right dimensions, reflecting the chosen sample sizes.

4.2.2 SIMULATING THE ECOLOGICAL PROCESS AND ITS OUTCOME: GREAT TIT ABUNDANCE

To simulate the abundance of great tits at each site, we choose values for the parameters that govern the spatial variation in abundance, β_0 to β_3. The first parameter is the average expected abundance, on the log scale, of great tits when all covariates have a value of zero—i.e., the intercept of the abundance model. We usually prefer thinking about tits in terms of their abundance, rather than their log(abundance). Hence, we choose the average abundance first and then link-transform.

```
mean.lambda <- 2                  # Mean expected abundance of great tits
beta0 <- log(mean.lambda)         # Same on log scale (= log-scale intercept)
beta1 <- -2                       # Effect (slope) of elevation
beta2 <- 2                        # Effect (slope) of forest cover
beta3 <- 1                        # Interaction effect (slope) of elev and forest
```

We apply the linear model and obtain the logarithm of the expected abundance of great tits, then use exponentiation to get the expected abundance of great tits, and plot everything (Figure 4.2).

```
log.lambda <- beta0 + beta1 * elev + beta2 * forest + beta3 * elev * forest
lambda <- exp(log.lambda)         # Inverse link transformation

par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
curve(exp(beta0 + beta1*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 18),
xlab = "Elevation", ylab = "lambda", lwd = 2)
text(-0.9, 17, "A", cex = 1.5)
plot(elev, lambda, frame.plot = FALSE, ylim = c(0, 38), xlab = "Elevation", ylab = "")
text(-0.9, 36, "B", cex = 1.5)
curve(exp(beta0 + beta2*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 18),
xlab = "Forest cover", ylab = "lambda", lwd = 2)
text(-0.9, 17, "C", cex = 1.5)
plot(forest, lambda, frame.plot = FALSE, ylim = c(0, 38), xlab = "Forest cover", ylab = "")
text(-0.9, 36, "D", cex = 1.5)
```

To better show the joint relationships between the expected abundance, elevation, and forest cover, we can compute the expected abundance of great tits for a grid spanned by a range of observed values for both covariates, and then visualize the 3-D relationship (Figure 4.3(a)). Note that by doing

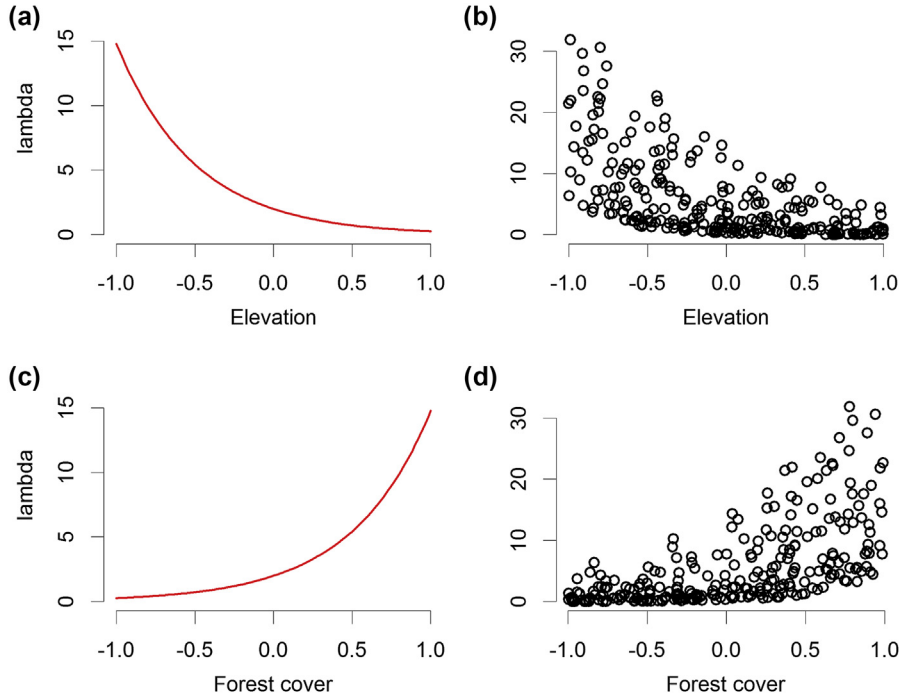

FIGURE 4.2

Two ways of showing the relationships between the expected abundance of great tits (lambda) and the two site covariates of elevation and forest cover. (a) Relationship of lambda–elevation for a constant value of forest cover (at the average of zero). (b) Relationship of lambda–elevation at the observed value of forest cover. (c) Relationship of lambda–forest cover for a constant value of elevation (at the average of zero). (d) Relationship of lambda–forest cover at the observed value of elevation.

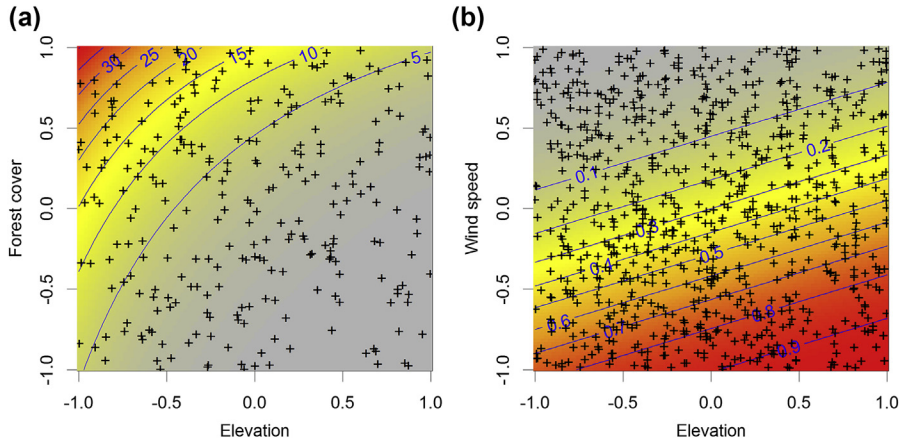

FIGURE 4.3

Relationships built into the simulated data between the expected abundance of great tits (`lambda`) and elevation and forest cover simultaneously (a), and between the expected detection probability of great tits (`p`) and elevation and wind speed simultaneously (b). Plus signs indicate the values of these covariates in the simulated data set. Note how the curvature of the contour lines in (a) indicates the presence of an interaction between the two covariates.

this, we are not changing anything with the simulated data; we simply execute this code to better visualize the relationships built into the simulated data between `lambda` (and `N` *and* `C`, below), `elev`, and `forest`.

```
# Compute expected abundance for a grid of elevation and forest cover
cov1 <- seq(-1,1,,100)                          # Values for elevation
cov2 <- seq(-1,1,,100)                          # Values for forest cover
lambda.matrix <- array(NA, dim = c(100, 100))   # Prediction matrix, for every combination
of values of elevation and forest cover
for(i in 1:100){
   for(j in 1:100){
      lambda.matrix[i, j] <- exp(beta0 + beta1 * cov1[i] + beta2 * cov2[j] + beta3 * cov1[i] *
cov2[j])
   }
}

par(mfrow = c(1, 2), mar = c(5,4,3,2), cex.main = 1.6)
mapPalette <- colorRampPalette(c("grey", "yellow", "orange", "red"))
image(x = cov1, y = cov2, z = lambda.matrix, col = mapPalette(100), xlab = "Elevation",
 ylab = "Forest cover", cex.lab = 1.2)
contour(x = cov1, y = cov2, z = lambda.matrix, add = TRUE, lwd = 1)
matpoints(elev, forest, pch="+", cex=0.8)      # add observed cov values
```

Thus, we see that the slope of the positive effect of forest cover, and the negative effect of elevation on the expected abundance, are modified by the positive interaction (Figure 4.3 (a)): when both elevation and forest cover have either low or high values, the expected abundance is greater than what we expect from the main effects of these covariates. Without interaction, the contours would be straight lines (as in Figure 4.3 (b)).

So far, we have not built any stochasticity into the relationships between great tit abundance and the covariates: this comes only with the adoption of some statistical model, or a statistical distribution, to describe the random variability around the expected value `lambda`. As a typical choice, we draw the actual number of tits at each site i, N_i, from a Poisson distribution with the given expectation (λ_i).

```
N <- rpois(n = M, lambda = lambda)          # Realized abundance
sum(N)                                      # Total population size at M sites
[1] 1507
table(N)                                    # Frequency distribution of tit abundance
N
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 61  41  27  19  14  16   7  14   5   6   5   5   5   5   3   4
 16  17  18  19  20  21 22  23  24  25  26  27  28  32  38
  3   2   2   5   6   1  2   1   1   1   1   1   1   2   1
```

We have now created the result of the ecological process: site-specific abundance, N_i. We see that 61 sites are unoccupied, and the remaining 206 sites have a total of 1507 great tits, with 1 to 38 individuals each.

4.2.3 SIMULATING THE OBSERVATION PROCESS AND ITS OUTCOME: POINT COUNTS OF GREAT TITS

Abundance N is not what we usually get to see; rather, there is always a chance of overlooking or failing to hear an individual. Hence, there is a binary measurement error when "measuring" abundance (see Section 1.4). We assume here that we can make only one of the two possible observation or measurement errors: we can fail to see an individual great tit that is there, and thus detection probability is less than 1, and measurement error is affected by elevation and wind speed. We never record an individual that is not there, or count the same individual multiple times, so we assume there are no false-positives. To make explicit that we could build an interaction effect between the two covariates into our data, we next allow for an interaction effect in the code, but set it to zero, and so effectively drop it from the model used for data generation. We first choose the values for α_0 to α_3, where the first is the expected detection probability of an individual great tit, on the logit scale, when all detection covariates have a value of zero. We choose the intercept of the detection model and then link-transform. The result of this is not the same as the average detection probability, which will actually be higher in our simulation; see below.

```
mean.detection <- 0.3                    # Mean expected detection probability
alpha0 <- qlogis(mean.detection)         # Same on logit scale (intercept)
alpha1 <- 1                              # Effect (slope) of elevation
alpha2 <- -3                             # Effect (slope) of wind speed
alpha3 <- 0                              # Interaction effect (slope) of elev and wind
```

Applying the linear model, we get the logit of the probability of detecting a great tit for each site and survey, and applying the inverse logit transformation, we get a matrix of dimension 267 by 3, with detection probability for each site i and survey j. Finally, we plot the relationships for detection probability in the data (Figure 4.4).

```
logit.p <- alpha0 + alpha1 * elev + alpha2 * wind + alpha3 * elev * wind
p <- plogis(logit.p)                     # Inverse link transform
mean(p)                                  # average per-individual p is 0.38

par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
curve(plogis(alpha0 + alpha1*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1.1),
xlab = "Elevation", ylab = "p", lwd = 2)
text(-0.9, 1.05, "A", cex = 1.5)
matplot(elev, p, pch = "*", frame.plot = FALSE, ylim = c(0, 1.1), xlab = "Elevation",
ylab = "")
text(-0.9, 1.05, "B", cex = 1.5)
curve(plogis(alpha0 + alpha2*x), -1, 1, col = "red", frame.plot = FALSE, ylim = c(0, 1.1),
xlab = "Wind speed", ylab = "p", lwd = 2)
text(-0.9, 1.05, "C", cex = 1.5)
matplot(wind, p, pch = "*", frame.plot = FALSE, ylim = c(0, 1.1), xlab = "Wind speed",
ylab = "p")
text(-0.9, 1.05, "D", cex = 1.5)
```

We similarly produce a plot of the joint relationships between elevation, wind speed, and detection probability of an individual great tit (Figure 4.3(b)). Without an interaction, the relationship on the link scale is represented by a plane with constant slopes in the x and y directions, respectively.

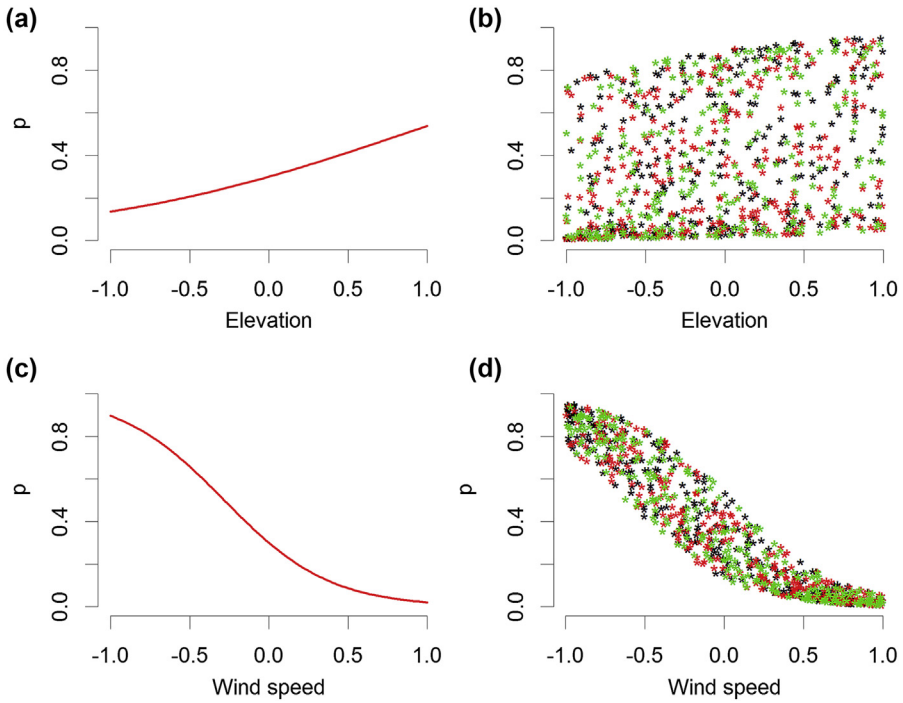

FIGURE 4.4

Two ways of depicting the relationships between the expected detection probability of an individual great tit (*p*) and the two covariates elevation and wind speed. (a) Relationship of *p*–elevation for a constant value of wind speed (at the average of zero). (b) Relationship of *p*–elevation at the observed value of wind speed. (c) Relationship of *p*–wind speed for a constant value of elevation (at the average of zero). (d) Relationship of *p*–wind speed at the observed value of elevation. In panels (b) and (d), colors represent different temporal replicates (surveys).

```
# Compute expected detection probability for a grid of elevation and wind speed
cov1 <- seq(-1, 1,,100)                    # Values of elevation
cov2 <- seq(-1,1,,100)                     # Values of wind speed
p.matrix <- array(NA, dim = c(100, 100)) # Prediction matrix which combines every value in
cov 1 with every other in cov2
for(i in 1:100){
   for(j in 1:100){
      p.matrix[i, j] <- plogis(alpha0 + alpha1 * cov1[i] + alpha2 * cov2[j] +
alpha3 * cov1[i] * cov2[j])
   }
}
image(x = cov1, y = cov2, z = p.matrix, col = mapPalette(100), xlab = "Elevation",
ylab = "Wind speed", cex.lab = 1.2)
contour(x = cov1, y = cov2, z = p.matrix, add = TRUE, lwd = 1)
matpoints(elev, wind, pch="+", cex=0.7, col = "black")  # covariate values in data set
```

When "measuring" abundance, imperfect detection represents a binomial measurement error mechanism; i.e., each great tit is either detected with probability p or not detected with probability 1-p. We apply this observation process now to produce replicate tit counts for each site.

```
C <- matrix(NA, nrow = M, ncol = J)     # Prepare array for counts
for (i in 1:J){                         # Generate counts
  C[,i] <- rbinom(n = M, size = N, prob = p[,i])
}
```

So here, at long last, are our simulated counts of great tits at 267 sites during three survey occasions! Let us look at them in tables and in a figure (Figure 4.5). Remember that sites are in the rows, and replicate surveys are in the columns. For comparison, we show the true abundance of tits in the first column.

```
head(cbind("True N" = N, "1st count" = C[,1], "2nd count" = C[,2], "3rd count" =
C[,3]), 10)                     # First 10 rows (= sites)
      True N   1st count   2nd count   3rd count
 [1,]      9           8           0           1
 [2,]      9           1           2           7
 [3,]      7           0           7           5
 [4,]      0           0           0           0
 [5,]      0           0           0           0
 [6,]      0           0           0           0
 [7,]      8           0           5           1
 [8,]      5           1           3           4
 [9,]      0           0           0           0
[10,]      1           0           1           0

table(C)
C
  0   1   2   3   4   5   6   7   8   9  10  11  12  13
408 142  69  38  39  30  18  12  13   7   2   3   4   3
 14  15  16  17  21  22  23  30
  4   1   2   1   1   2   1   1

par(mfrow = c(2, 2), mar = c(5,4,2,2), cex.main = 1)
matplot(elev, C, pch = "*", frame.plot = FALSE, ylim = c(0, 38), xlab = "Elevation",
ylab = "Count (C)")
text(-0.9, 36, "A", cex = 1.5)
matplot(forest, C, pch = "*", frame.plot = FALSE, ylim = c(0, 38), xlab = "Forest cover", ylab
= "Count (C)")
text(-0.9, 36, "B", cex = 1.5)
matplot(wind, C, pch = "*", frame.plot = FALSE, ylim = c(0, 38), xlab = "Wind speed",
ylab = "Count (C)")
text(-0.9, 36, "C", cex = 1.5)
hist(C, breaks = 50, col = "grey", ylim = c(0, 460), main = "", xlab = "Count (C)")
text(3, 450, "D", cex = 1.5)
```

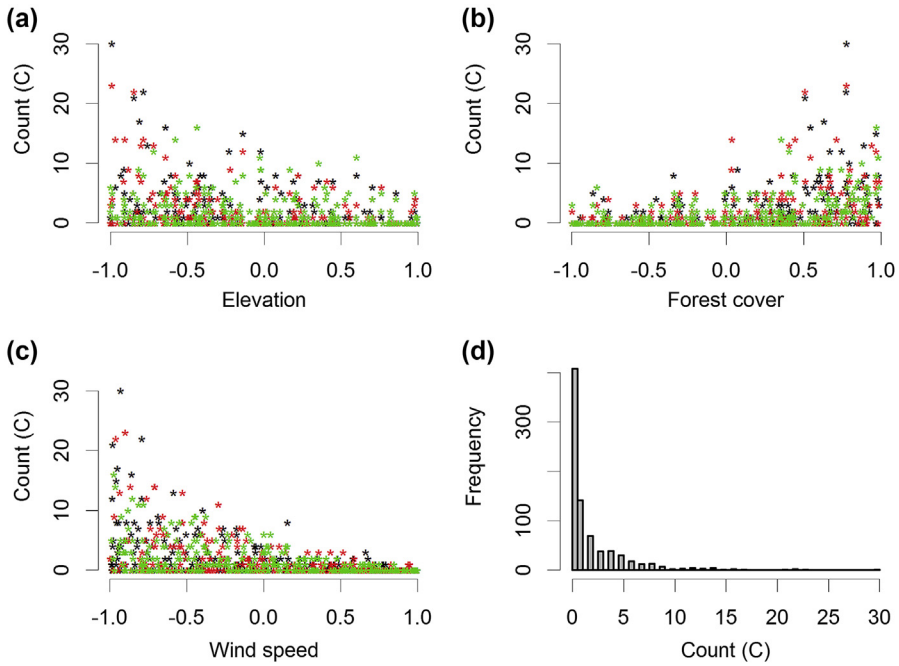

FIGURE 4.5

Relationships between the observed counts of great tits (*C*) and the three scaled covariates elevation (a), forest cover (b), and wind speed (c); and frequency distribution of the observed counts in the simulated data set for 267 sites with three surveys each (d). Different colors in a–c represent different temporal replicates (surveys).

Thus, we have created a data set where counts of great tits *C* are negatively related to elevation and wind speed and positively related to forest cover. The reason for these covariate relationships is fundamentally different. Site abundance, the logical target of ecological inference, is affected by forest cover and elevation, but *not* by wind speed, while detection probability, the parameter characterizing the measurement error process when taking measurements of abundance, is also affected by elevation, but by wind speed, too. Hence, we see that it may be challenging to disentangle the reasons for spatiotemporal variation in observed *counts*, since they can be affected by two entirely different processes: ecological and observational.

We can characterize the effects of imperfect detection in two ways: on perceived abundance and on perceived occurrence. The maximum count at each site is an observed measure for site-specific abundance of great tits, *N*. Hence, the sum of the maximum tit counts over sites is a readily available "detection-naïve" estimate of total abundance for all 267 sites combined.

```
sum(N)                  # True total abundance (all sites)
sum(apply(C, 1, max))   # 'Observed' total abundance (all sites)
[1] 1507
[1] 871
```

Thus, the combined estimation error for total great tit abundance of a procedure that does not model imperfect detection, but treats the counts as estimates of abundance, is $(1507-871)/1507 = 0.42$, representing an underestimation of great tit population size of 42%.

Occurrence or "presence," the "unit" of species distribution studies, denotes the case that $N_i > 0$ (i.e., that abundance at site i is greater than zero). Here are the corresponding figures for occurrence:

```
sum(N>0)                  # True number of occupied sites
sum(apply(C, 1, max)>0)   # 'Observed' number of occupied sites
[1] 206
[1] 190
```

Hence, in terms of great tit occurrence, the error in the total species distribution induced by imperfect detection in the sample of studied sites is only $(206-190)/206$, or about 8%, much less than the total error for the abundance measurement.

In later chapters, you will see hierarchical models that allow you to disentangle the parameters governing the ecological process from those of the observation process, especially in Chapter 6 where we cover the binomial *N*-mixture model (Royle, 2004b). This model is represented exactly by the two processes that we have assumed when simulating this data set. But this is not what we do first; for now, we generated this data set for three reasons: first, to illustrate how data sets can be simulated; second, to describe the two processes underlying all abundance measurements in nature; and third, to generate an example data set with which to introduce the Bayesian modeling software WinBUGS, OpenBUGS, and JAGS in Chapter 5.

4.3 PACKAGING EVERYTHING IN A FUNCTION

It can be very useful to package a simulation that you run repeatedly into a function. This will make your programming more concise and flexible, and it makes more transparent the settings used for data generation. Therefore, here we define a function to generate the same kind of data that we just created, assigning function arguments to any part of the simulation code that we might want to flexibly modify among simulated data sets, such as sample size, parameter values, presence/absence/magnitude of interaction terms, or detection error.

```
# Function definition with set of default values
data.fn <- function(M = 267, J = 3, mean.lambda = 2, beta1 = -2, beta2 = 2, beta3 = 1,
mean.detection = 0.3, alpha1 = 1, alpha2 = -3, alpha3 = 0, show.plot = TRUE){
#
# Function to simulate point counts replicated at M sites during J occasions.
# Population closure is assumed for each site.
# Expected abundance may be affected by elevation (elev),
# forest cover (forest) and their interaction.
# Expected detection probability may be affected by elevation,
# wind speed (wind) and their interaction.
# Function arguments:
#     M: Number of spatial replicates (sites)
```

```
#     J: Number of temporal replicates (occasions)
#     mean.lambda: Mean abundance at value 0 of abundance covariates
#     beta1: Main effect of elevation on abundance
#     beta2: Main effect of forest cover on abundance
#     beta3: Interaction effect on abundance of elevation and forest cover
#     mean.detection: Mean detection prob. at value 0 of detection covariates
#     alpha1: Main effect of elevation on detection probability
#     alpha2: Main effect of wind speed on detection probability
#     alpha3: Interaction effect on detection of elevation and wind speed
#     show.plot: if TRUE, plots of the data will be displayed;
#        set to FALSE if you are running simulations.

# Create covariates
elev <- runif(n = M, -1, 1)                              # Scaled elevation
forest <- runif(n = M, -1, 1)                            # Scaled forest cover
wind <- array(runif(n = M *J, -1, 1), dim = c(M, J))     # Scaled wind speed

# Model for abundance
beta0 <- log(mean.lambda)              # Mean abundance on link scale
lambda <- exp(beta0 + beta1*elev + beta2*forest + beta3*elev*forest)
N <- rpois(n = M, lambda = lambda)     # Realised abundance
Ntotal <- sum(N)                       # Total abundance (all sites)
psi.true <- mean(N>0)                  # True occupancy in sample

# Plots
if(show.plot){
par(mfrow = c(2, 2), cex.main = 1)
devAskNewPage(ask = TRUE)
curve(exp(beta0 + beta1*x), -1, 1, col = "red", main = "Relationship lambda-elevation
\nat average forest cover", frame.plot = F, xlab = "Scaled elevation")
plot(elev, lambda, xlab = "Scaled elevation", main = "Relationship lambda-elevation
\nat observed forest cover", frame.plot = F)
curve(exp(beta0 + beta2*x), -1, 1, col = "red", main = "Relationship lambda-forest \ncover
at average elevation", xlab = "Scaled forest cover", frame.plot = F)
plot(forest, lambda, xlab = "Scaled forest cover", main = "Relationship lambda-forest
cover \nat observed elevation", frame.plot = F)
}

# Model for observations
alpha0 <- qlogis(mean.detection)       # Mean detection on link scale
p <- plogis(alpha0 + alpha1*elev + alpha2*wind + alpha3*elev*wind)
C <- matrix(NA, nrow = M, ncol = J)    # Prepare matrix for counts
for (i in 1:J){                        # Generate counts by survey
  C[,i] <- rbinom(n = M, size = N, prob = p[,i])
}
summaxC <- sum(apply(C,1,max))         # Sum of max counts (all sites)
psi.obs <- mean(apply(C,1,max)>0)      # Observed occupancy in sample
```

```
# More plots
if(show.plot){
par(mfrow = c(2, 2))
curve(plogis(alpha0 + alpha1*x), -1, 1, col = "red", main = "Relationship p-elevation \nat
average wind speed", xlab = "Scaled elevation", frame.plot = F)
matplot(elev, p, xlab = "Scaled elevation", main = "Relationship p-elevation\n at observed
wind speed", pch = "*", frame.plot = F)
curve(plogis(alpha0 + alpha2*x), -1, 1, col = "red", main = "Relationship p-wind speed \n
at average elevation", xlab = "Scaled wind speed", frame.plot = F)
matplot(wind, p, xlab = "Scaled wind speed", main = "Relationship p-wind speed \nat
observed elevation", pch = "*", frame.plot = F)

matplot(elev, C, xlab = "Scaled elevation", main = "Relationship counts and elevation",
 pch = "*", frame.plot = F)
matplot(forest, C, xlab = "Scaled forest cover", main = "Relationship counts and forest
cover", pch = "*", frame.plot = F)
matplot(wind, C, xlab = "Scaled wind speed", main = "Relationship counts and wind speed",
pch = "*", frame.plot = F)
desc <- paste('Counts at', M, 'sites during', J, 'surveys')
hist(C, main = desc, breaks = 50, col = "grey")
}

# Output
return(list(M = M, J = J, mean.lambda = mean.lambda, beta0 = beta0, beta1 = beta1, beta2 =
beta2, beta3 = beta3, mean.detection = mean.detection, alpha0 = alpha0, alpha1 = alpha1,
alpha2 = alpha2, alpha3 = alpha3, elev = elev, forest = forest, wind = wind, lambda = lambda,
N = N, p = p, C = C, Ntotal = Ntotal, psi.true = psi.true, summaxC = summaxC, psi.obs =
psi.obs))
}
```

Once we have defined the function by executing the code of its definition in R, we can call the function repeatedly and send its results to the display, or more commonly assign them to an R object so that we can use the generated data set in an analysis.

```
data.fn()                                          # Execute function with default arguments
data.fn(show.plot = FALSE)                         # same, without plots
data.fn(M = 267, J = 3, mean.lambda = 2, beta1 = -2, beta2 = 2, beta3 = 1, mean.detection =
0.3, alpha1 = 1, alpha2 = -3, alpha3 = 0)          # Explicit defaults
data <- data.fn()                                  # Assign results to an object called 'data'
```

Perhaps the simplest possible use of this function is to experience sampling error: the natural variability of repeated realizations (i.e., data sets) from our stochastic process, or of things calculated from these data sets. Let us simulate 10,000 data sets of great tit counts and see how they vary in terms of the true total population size of tits (`Ntotal`), and of the next-best observable thing: the sum of the max counts of tits (`summaxC`).

```
simrep <- 10000
NTOTAL <- SUMMAXC <- numeric(simrep)
```

```
for(i in 1:simrep){
   data <- data.fn(show.plot = FALSE)
   NTOTAL[i] <- data$Ntotal
   SUMMAXC[i] <- data$summaxC
}
plot(sort(NTOTAL), ylim = c(min(SUMMAXC), max(NTOTAL)), ylab = "", xlab = "Simulation",
col = "red", frame = FALSE)
points(SUMMAXC[order(NTOTAL)], col = "blue")
```

We see that different realizations from the identical stochastic process, as represented by the relations defined in the data-generating function, can yield dramatically different outcomes (Figure 4.6). Among the 10,000 simulation replicates, the total population size varied from about 1000 to 1850, and the sum of the maximum counts varied from 600 to 1100. On average, the ratio of the two was 58%, but varied from 50%–67%. This ratio is a measure of the detection probability over the three surveys combined; hence, `summaxC` clearly is not an unbiased estimator of `Ntotal` (for a finite number of surveys). We also see that for any given total population size *N*, the sum of the max counts varied by almost 200 among replicate data sets. In real life, we are typically faced with a single realization of a stochastic process such as this one, and we aim to estimate its key descriptors—i.e., the parameters in the statistical model. Seeing how much this can vary from one realization of a single process to the next, we understand that this can be a daunting task!

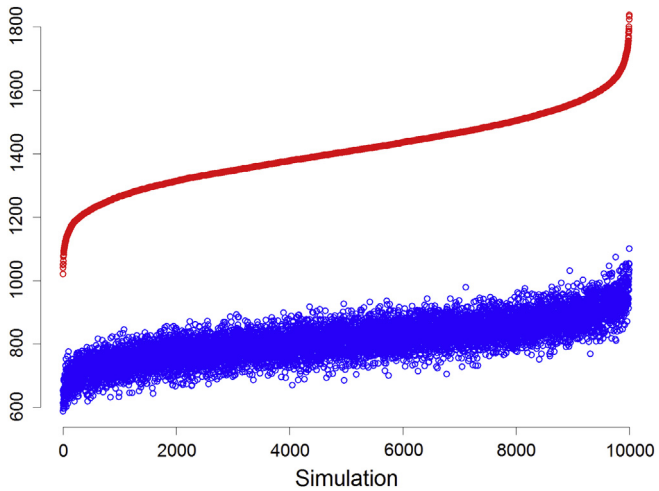

FIGURE 4.6

Natural variability (i.e., sampling error) of total population size (`Ntotal`; ordered by size, in red) and of the sum of the maximum counts over all sites (`summaxC`, in blue) in a simulated great tit population. The latter is a detection-naïve (observable) estimator of total population size. The figure shows the result of 10,000 calls to function `data.fn`.

You can use this function to generate data sets under various sampling designs (e.g., number of sites or surveys) and for a variety of ecological and sampling situations (i.e., patterns in abundance and detection).

```
data.fn(J = 2)                          # Only 2 surveys
data.fn(J = 1)                          # No temporal replicate
data.fn(M = 1, J = 100)                 # No spatial replicates, but 100 counts
data.fn(beta3 = 1)                      # With interaction elev-wind on p
data.fn(M = 267, J = 3, mean.lambda = 2, beta1 = -2, beta2 = 2, beta3 = 1, mean.detection = 1)
                                        # No obs. process (i.e., p = 1, perfect detection)
data.fn(mean.lambda = 50)               # Really common species
data.fn(mean.lambda = 0.05)             # Really rare species
```

On every function execution, we get a different realization from the random process defined in the function. To get the same data set every time, we can set a random number seed.

```
set.seed(24)
data <- data.fn()                       # Default arguments
str(data)                               # Look at the object
List of 23
 $ M                 : num 267
 $ J                 : num 3
 $ mean.lambda       : num 2
 $ beta0             : num 0.693
 $ beta1             : num -2
 $ beta2             : num 2
 $ beta3             : num 1
 $ mean.detection    : num 0.3
 $ alpha0            : num -0.847
 $ alpha1            : num 1
 $ alpha2            : num -3
 $ alpha3            : num 0
 $ elev              : num [1:267] -0.4149 -0.5502 0.4084 0.0378 0.3252 ...
 $ forest            : num [1:267] 0.432 0.446 0.97 -0.894 -0.278 ...
 $ wind              :num [1:267, 1:3] -0.8395 -0.0811 0.6636 -0.7518 -0.9838 ...
 $ lambda            : num [1:267] 9.093 11.474 9.129 0.3 0.547 ...
 $ N                 : int [1:267] 9 9 7 0 0 0 8 5 0 1 ...
 $ p                 : num [1:267, 1:3] 0.7784 0.2397 0.0809 0.8094 0.919 ...
 $ C                 : int [1:267, 1:3] 8 1 0 0 0 0 0 1 0 0 ...
 $ Ntotal            : int 1507
 $ psi.true          : num 0.772
 $ summaxC           : int 871
 $ psi.obs           : num 0.712
```

To make the objects inside the list directly accessible to R, without having to address them as `data$C` for instance, you can attach `data` to the search path.

```
attach(data)     # Make objects inside of 'data' accessible directly
```

Remember to detach the data after use, and in particular before attaching a new data object, because more than one data set attached in the search path will cause confusion.

```
detach(data)      # Make clean up
```

4.4 SUMMARY AND OUTLOOK

The two main purposes served by this chapter were: to introduce data simulation and to reinforce, using R code, what we perceive as the two typical processes underlying *all* count data in ecology: an ecological process and an observation, or measurement, process. By data simulation, we understand the generation of a data set as the random outcome from a defined stochastic process. We described it by statistical distributions with effects of covariates built into the processes in the manner of GLMs. We have given an extended version of this code, where each step was explained and commented, and an abbreviated version where the key lines are packaged into a function with arguments that can be chosen upon calling the function. Using functions is often how we conduct data simulation, but we wanted to provide the extended version first to better illustrate the great importance of data simulation for applied statistics. We have pointed out some of the important benefits of data simulation.

- When we analyze a data set using a model similar to the one used for data simulation, we can compare the resulting inferences with the known truth. This can help avoid coding errors in the analysis, or allow us to diagnose problems with the MCMC algorithms. We will see this throughout the book.
- We have just seen how we can observe sampling error by repeatedly executing the data simulation code: unless using a seed, every single data set will differ, and so will things calculated from the observed data, such as mean counts or the significance of a parameter estimate. Sampling error can be observed by plotting or summarizing the distribution of those variables of interest across simulated data sets.
- As we will see in 6.6 for *N*-mixture models, and in 10.7 for site-occupancy models, we can check whether some estimator (for instance, the MLE or the posterior mean) is unbiased and gauge its precision for various sample sizes or magnitudes of measurement errors. In a similar vein, we have just seen that in the presence of imperfect detection, the sum of the maximum counts over a number of sites is not an unbiased estimator of the total abundance at these sites with three surveys. (However, it will become increasingly unbiased with increasing number of replicate surveys.)
- We can do power analyses by setting a parameter of interest at a particular value and creating and analyzing replicate data sets. Power is then estimated by the proportion of times that the parameter estimate is significant at a chosen significance level. Repeating this for different sample sizes such as number of sites or surveys can give information relevant for the design of a study.
- We can check the identifiability of a parameter.
- We can check estimator robustness and gauge the effects of assumption violations. For instance, in this chapter, we could generate data including an interaction between wind and elevation on detection, and then analyze the resulting data without the interaction to find out how robust estimates of the main effects are to omission of an existing interaction. In Chapter 6 we will see how the *N*-mixture model is affected by the presence of unmodeled effects, and in Chapter 16

(in volume 2), we will see how unmodeled detection heterogeneity affects the parameter estimates in a dynamic occupancy model.

- And finally, we repeat that if you *really* understand the data simulation under a specific model, you then understand that model. Hence, if you truly understand the simulation in this chapter, then you also understand the basic *N*-mixture model (Royle, 2004b) that is the subject of Chapter 6.

Just as we value the confirmation of field studies with lab experiments, we also believe that the benefits of simulating data are so great that almost any major data analysis project should be complemented with some simulation studies. Nevertheless, some may argue against the use of simulated data sets: First, ecologists often appear to immediately become less interested when they know a data set is simulated rather than "real," and second, simulated data may be "too simple" compared with real data sets. Of course, we are not saying you should *only* analyze simulated data sets; we see them as providing you with practice to be better able to manage the challenges of your real data sets. One way in which simulated data in this book are simpler than those of real data sets is that we usually do not create missing values. Rather, we generate balanced data sets; for instance, in this chapter we assumed the same number of replicate counts at every site. Of course, when you think that a certain complication such as missing data is needed in your analysis, you should simply build it into your data-generating procedure. For instance, when you worry that uneven numbers of counts among sites may affect the quality of your estimates, you could simply turn some of your data into missing values to mimic a pattern in which there are uneven numbers of counts per site, or even sites without counts. Then you can analyze these data sets to see whether there is an effect of a certain number or pattern of missing values on the estimates.

Simulating a data set has permitted us to illustrate the salient features of ecological count data: they are generated by an ecological process and an observation process. We like to call the latter a *measurement error process* because that is exactly what it is: it introduces error into our measurement C of abundance N. Although there are two possible kinds of measurement errors for counts (false-positives and false-negatives), we have only included the more common false-negative error for now, where the error rate is given by the complement of detection probability. Almost all models introduced in later chapters accommodate this type of measurement error, but some (see Chapter 19) will also accommodate false-positives. Incorporating false-positives is an active area of research in capture–recapture modeling.

Throughout this book, we will see many more examples of these topics, both data simulation, and the interplay of an ecological and a measurement error process underlying observed data on distribution and abundance. Some of these R functions will be considerably more complex and can be used to generate an even wider variety of data sets than the fairly simple function in this chapter. Data simulation is a vast field, and we have only shown a very simple example: you may want to use much more complicated ways of simulating data, either to analyze the resulting data sets or to study emerging properties of a system. One important field for the latter are so-called individual- or agent-based models (e.g., Grimm, 1999; Railsback and Grimm, 2012).

In Chapter 5, we use our function to generate a data set of counts of great tits, and analyze it to illustrate the fitting of several GLMs and simple random-effects models using WinBUGS, OpenBUGS, and JAGS, as well as using standard or restricted maximum likelihood procedures in R. Then, in Chapter 6, we will analyze exactly this type of data using the hierarchical *N*-mixture model, and we will encounter a much more complex version of a data simulation function

that we will use in Chapters 6 and 7. If you understood the data simulation procedure in this chapter, then you fully understand the *N*-mixture model, which is a cornerstone of the hierarchical models in this book.

EXERCISES

1. *Sampling error*: Simulate count data with $p = 1$ and a single temporal replicate (and default function arguments otherwise). Fit the data-generating model using R function `glm`, and observe how variable the estimates are (perhaps run 1000 simulation replicates). This is sampling error and is quantified by the standard error of the estimates.
2. *Small-sample bias*: Repeat Exercise 1, but with a very small sample size (e.g., 5 or 10 sites), and inspect bias and precision of the maximum likelihood regression estimators.
3. *Power analysis*: With default arguments apart from $p = 1$ and a single survey per site, what is the power to detect the effects in abundance using a Poisson GLM fit with `glm`? Repeat with 20 sites and with 200 sites.
4. *Effects of ignoring the observation process*: Simulate data sets using the default arguments (but only one survey), and analyze them with a Poisson GLM with the data-generating structure in abundance. What is the effect of ignoring the measurement error (i.e., the observation process)?
5. *"Reliability" of count data (1)*: Run the data simulation function with a constant detection probability of 0.5 (i.e., set the coefficients for elevation and wind to zero). Does a constant measurement error of $(1-p) = 0.5$ standardize the counts? Will we always count the same number of tits at a particular site? Why/why not? What can be done to make the counts more reliable in the sense of "less variable"?
6. *"Reliability" of count data (2)*: Continuing, are replicated counts that are less variable more or less reliable as an index to the local population size *N*? Plot the SD of counts obtained with $p = 0, 0.1, \ldots, 0.9, 1$.
7. *When is the observed maximum count an unbiased estimator of abundance?* By trial and error, check how many surveys are required in the simulation at the end of Section 4.3 to make `summaxC` an unbiased estimator of `Ntotal`. Be prepared to wait for a while.
8. *Relationship of detection/nondetection data to counts and of cloglog-Bernoulli model to Poisson regression*: In 3.3.6, the Bernoulli model with complementary log-log link did not produce adequate estimates from the mere detection/nondetection data about the relationship between the actual mite counts and the covariates. We have claimed that this is probably due to the tiny sample size and serious departure from a Poisson. See whether the cloglog-Bernoulli model provides better estimates of an abundance relationship underlying detection/nondetection data for a large sample size, when the count data are in fact Poisson. Use the data simulation function in this chapter for 267 sites with a single replicate and perfect detection.
9. *Sensitivity of trends based on counts vs. trends based on detection/nondetection (presence/absence) data (see also Pollock, 2006)*: Simulate a population that declines at a chosen rate over a selected number of years. You can use parts of the simulation function and ignore the observation process (i.e., assume $p = 1$). Estimate the power to detect a population decline of your choice and for a selected number of years when you use the counts (you can use 1000 simulation reps). Then, do the same for detection/nondetection data by collapsing the counts to

0 and 1 and fitting a logistic regression. Fit both models for each simulated data set to enhance comparability (this is a more challenging exercise).

10. *Generation of overdispersion*: The variance of a Poisson random variable is equal to its mean. Use the data simulation function with the effects of all covariates set to 0, and mean detection to 1, in order to verify this. Then, add in first one, then two, and finally all three abundance covariates (keeping detection perfect all the time). Observe how the variance/mean ratio of the counts changes. What does this mean? Are these data no longer Poisson-distributed?

11. *Data simulation for hierarchical detection/nondetection data*: Modify the function such that you generate detection/nondetection (or "presence/absence") instead of abundance data, according to a site-occupancy model (see Chapter 10) with logit(psi) = covariate effects, and logit(detection) = covariate effects (see also Section 10.5).