

# BAYESIAN POPULATION ANALYSIS USING WinBUGS

---

# BAYESIAN POPULATION ANALYSIS USING WinBUGS

---

## A HIERARCHICAL PERSPECTIVE

MARC KÉRY AND MICHAEL SCHAUB

Swiss Ornithological Institute  
6204 Sempach  
Switzerland

*Foreword by*

STEVEN R. BEISSINGER



AMSTERDAM • BOSTON • HEIDELBERG • LONDON  
NEW YORK • OXFORD • PARIS • SAN DIEGO  
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Academic Press is an imprint of Elsevier



Academic Press is an imprint of Elsevier  
225 Wyman Street, Waltham, MA 02451, USA  
525 B Street, Suite 1900, San Diego, CA 92101-4495, USA  
The Boulevard, Langford Lane, Kidlington, Oxford, OX51GB, UK  
Radarweg 29, PO Box 211, 1000 AE Amsterdam, The Netherlands

First edition 2012

Copyright © 2012 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone (+44) (0) 1865 843830; fax (+44) (0) 1865 853333; e-mail: [permissions@elsevier.com](mailto:permissions@elsevier.com). Alternatively you can submit your request online by visiting the Elsevier Web site at <http://elsevier.com/locate/permissions>, and selecting *Obtaining permission to use Elsevier material*.

#### Notice

No responsibility is assumed by the publisher for any injury and/or damage to persons or property as a matter of products liability, negligence, or otherwise or from any use or operation of any methods, products, instructions, or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

#### Library of Congress Cataloging-in-Publication Data

Application submitted

#### British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library.

ISBN: 978-0-12-387020-9

For information on all Academic Press publications  
visit our Web site at [www.elsevierdirect.com](http://www.elsevierdirect.com)

*Typeset by:* diacriTech, Chennai, India

Printed and bound in China

12 13 14 15 10 9 8 7 6 5 4 3 2 1

Working together to grow  
libraries in developing countries

[www.elsevier.com](http://www.elsevier.com) | [www.bookaid.org](http://www.bookaid.org) | [www.sabre.org](http://www.sabre.org)

ELSEVIER

BOOK AID  
International

Sabre Foundation

*We dedicate this book to our children Gabriel, and Lilly and Lukas.*

# Foreword

---

Scientific disciplines are often judged by their success in translating ideas and principles into outcomes under future conditions. One of the greatest challenges faced by ecologists and conservation biologists of our time is to use our knowledge to understand how species, populations, and communities will behave, persist, and evolve in the future in a world with more people and a changing climate. Models are central to this undertaking.

Models have become an important tool for conservation planning and managing natural resources. In ecology and conservation biology as in other sciences, models have always driven the development of certain concepts and served a useful role in synthesizing knowledge and guiding research. Computer models also provided ways to gain new insights into modeled systems by running “virtual experiments.” But now more than ever, mathematical and simulation models are being used to project future outcomes based on past, current, or projected conditions. Over the past 30 years, models have grown in use, prominence, and complexity with the advent of desktop computers that have become both more affordable and more powerful to run them and with the growth of specialized software to enable users to implement them.

Models are both familiar to us and scary. We unconsciously use models everyday in making life choices. For example, we often use simple “rule of thumb” models when we dress to determine which color combinations are complementary and which are clashing, and we take projections from complex weather models into consideration when choosing whether to wear a warm or cool fabric. Scientists have routinely used conventional statistical models or “frequentist models” when determining whether relationships differ by testing a null hypothesis to a specified confidence level (e.g.,  $p < 0.05$ ). Yet, models are not a panacea. Many ecologists, conservation biologists, and resource managers distrust models because they can be overly complex, use mathematical methods, and contain computer code that they do not understand, or they are based on uncertain relationships and parameter estimates. The accepted convention of a  $p$  value of 0.05 (or a 5% chance of wrongly rejecting the null hypothesis of no difference when it is true) is an artificial construct and perhaps too restrictive for gleaning useful information from nature to use in management.

Bayesian methods can address some of these concerns. They use data or hypothesized relationships about data to make inferences about ecological systems. Bayesian models have the advantage of making probabilistic

statements about the veracity of hypotheses or relationships, given the data. They can explicitly incorporate uncertainty in model structure and parameter estimates through both prior knowledge and expectations about data into models that produce posterior distributions of outcomes. Bayesian methods depend on resampling distributions, and their recent growth is a result of both their utility and the ease with which computers can implement numerical recipes using Markov Chain Monte Carlo (MCMC) sampling.

This book provides an accessible introduction to Bayesian methods as applied to analyzing populations. It covers a breadth of applications that are widely used in ecology and population management, from analysis of count data and demographic rates for understanding the fluctuations of single populations to estimation of patch occupancy, and metapopulation dynamics that characterize more widely distributed species. Moreover, it uses a practical approach to model building that recognizes most data obtained in field studies of population ecology will have associated sampling uncertainties that arise from hidden processes, so-called hierarchical models. These models account for both the ecological processes of interest and the additional uncertainty caused by unobserved processes that always accompany field sampling, such as variation in the detectability of individuals. The book also makes extensive use of the free and widely used computer programs R and WinBUGS to implement these models. It is the ideal combination for both beginning students and beginning modelers to learn these methods. Advanced users will find plenty of wisdom in these pages to gain new skills, as I have.

Wise use of a model to make decisions that prevent extinction and recover populations requires understanding the unique attributes of a model, determining whether the assumptions that underlie the model's structure are valid and testing the ability of the model to predict the future correctly. This book goes a long way toward building models that can address the first two goals. Ecologists and conservation biologists will still have much work to do to determine how well their models perform. Although the future remains unpredictable as always, there will undoubtedly be a great need to manage wildlife and plant populations by applying the kinds of models presented in this book and by conducting field studies to improve their performance, if the looming extinctions that are projected to be associated with growing human populations and a warming climate are to be prevented.

*Steven R. Beissinger  
Berkeley, California*

# Preface

---

You are looking at a gentle introduction for ecologists to Bayesian population analysis using the BUGS software. We emphasize learning by doing and leisurely walk you through a wide range of statistical methods for a broad array of model classes that are relevant for population ecologists. We focus on hierarchical models for estimation and modeling of quantities such as population size or survival probability, while accounting for imperfect detection probability. The reading is intended to be light and engaging, while at the same time, we hope that the content is represented accurately.

This book has been written by ecologists for ecologists. For this project, two experienced population ecologists have teamed up in a complementary way. Marc has been working chiefly in projects involving the estimation and modeling of population size and occurrence and the simplest description of population dynamics, population trends. The work of Michael has focused on teasing apart the demographic rates that underlie the observed dynamics (i.e. trends) of a population. In this way, we neatly combine our strengths and experience. We have published papers that use WinBUGS to fit almost all model classes described in this book and have experience in their frequentist analysis as well.

Both in content and in style, this book is a sequel to a similar book written by Marc (Kéry, 2010). In content, the latter is more introductory and directed to the general ecologist or indeed to anybody interested in regression modeling in WinBUGS. In Kéry (2010), most of the typical ecological statistics examples, such as estimation of population size and demographic rates, that is, our Chapters 6–11, are lacking conspicuously. In addition, in Chapters 12 and 13, we now extend the binomial mixture and the site-occupancy model to multiple “seasons” and the single-state site-occupancy model to multiple states. These important generalizations are lacking in Kéry (2010). We make occasional reference to Kéry (2010), but our current book is independent from the earlier one. Nevertheless, should you find some material in here difficult to follow, we suggest to use Kéry (2010) for learning about ecological modeling in WinBUGS at a more introductory level.

In style, the key concepts of Kéry (2010) have been retained for this book:

1. We provide a large number of richly commented worked examples to illustrate a wide range of statistical models that are relevant to

- the research of a population ecologist and to the analyses of wildlife or fisheries managers or analysts in more applied branches of ecology.
- 2. We have written the book using WinBUGS, but most of the code should run fine with OpenBUGS and JAGS as well.
  - 3. All WinBUGS analyses are run from within software R; hence, this is also an R book.
  - 4. We provide a complete documentation of *all* R and WinBUGS code required to conduct *all* our analyses and show *all* the necessary steps from having the data in some sort of text file to interpreting and processing the output from WinBUGS in R. Thus, you are almost *guaranteed* to be able to replicate our analyses for your own data sets.
  - 5. We make extensive use of the simulation of data sets and their analysis. We believe that simulating data sets can be crucial to your understanding of the models. However, we also provide 1–2 analyses of real-life data in each chapter.
  - 6. We have a clear and consistent layout for all computer code.
  - 7. We aim at a light and engaging language.
  - 8. Each chapter has a set of exercises with solutions for all of them provided on the book website ([www.vogelwarte.ch/bpa](http://www.vogelwarte.ch/bpa)).

In scope and in style, our book intends to build a bridge between introductory texts by McCarthy (2007) or Kéry (2010) and three more advanced texts on the analysis of populations, metapopulations, and communities, which have recently been published and which all use WinBUGS as their primary software: Royle and Dorazio (2008), King et al. (2010), and Link and Barker (2010). If your primary research topic is population ecology as covered in our book, you should consider buying some or all of these books as well.

Our book is based on a one-week course for graduate students and post-doctoral researchers that we teach at universities and research institutes. For this course, we require participants to have some basic knowledge of program R or other programming languages as well as of basic statistical methods such as regression and ANOVA. It helps a lot if they have also had some exposure to generalized linear models (GLMs) and random-effects models and know what the design matrix of a linear model is. These requirements fairly accurately describe the intended audience of our book. We believe that our book is well suited for a one-semester course in modern population analysis for subjects such as quantitative conservation biology, resource management, fisheries, wildlife management, or general population ecology. In addition, our book is perfect for self-study, owing to its gentle style and because the complete code is shown and is amply documented. Our book website contains a text file with all R-WinBUGS code, data sets, solutions to all exercises, our utility functions,

additional bonus material, a list of Errata plus some other information, such as about upcoming workshops.

Recently, the active software development of the BUGS project has moved over from WinBUGS to OpenBUGS (Lunn et al., 2009 see [www.openbugs.info](http://www.openbugs.info)). As of early 2011, the syntax of the two BUGS sisters has remained virtually identical. We have written and tested our code in WinBUGS 1.4. (and with R 2.12.), but we have checked a sample in OpenBUGS also and most ran fine. The latest release of OpenBUGS contains a series of ecological examples that are all of relevance for the readers of this book. The JAGS software (see [www-fis.iarc.fr/~martyn/software/jags](http://www-fis.iarc.fr/~martyn/software/jags)) is another MCMC engine that uses the BUGS language, as do WinBUGS and OpenBUGS. Hence, most code in our book should run in JAGS as well. In contrast to Win- and OpenBUGS, JAGS also runs on Macs.

Here are a few tips on how to use this book. We strongly suggest you first read Chapters 1–4 because they contain important introductory material that you will need to know in later chapters. Only then should you pick chapters according to your interests. Evidently, before starting to work through this book, you need to have installed the necessary software: R, with some packages (especially R2WinBUGS, but also lme4) and WinBUGS, with both the upgrade patch and the immortality key decoded, or else have OpenBUGS or JAGS functional. When using WinBUGS from R, you need to always first load the R2WinBUGS package (Sturtz et al., 2005). We do not usually say this, but simply assume that you issue the command `library(R2WinBUGS)` at the start of every R session. In addition, you need to tell R where the WinBUGS executable is residing on your computer. For that, we define an object that contains this address (`bugs.dir <- "c:/Program Files/WinBUGS14/";` this is the default) and refer to it when calling WinBUGS with function `bugs()`. If WinBUGS is placed in another folder on your computer, the path information needs to be modified accordingly. Such information can also be written into the text file `Rsite.profile`, which sits in the R folder `etc` and contains global R settings (see Kéry, 2010, p. 32). Several models in the book take a long time to fit, hence, we give approximate bugs run times (BRT) for each. We use the R function `sink()` to write into the R working directory (which you can set yourself using `setwd()`), a text file containing the model description in the BUGS language. We find it useful to have all our code in a single document. You have to be totally clear about which part of the code is in the BUGS language and which is in the R language. This may be a little confusing at first, especially, because the two languages are quite similar (R is a dialect of S, and BUGS is strongly inspired by S). See the WinBUGS tips in Appendix 1 for more explanation. Finally, an important tip for when you cannot follow an analysis in this book is to execute code line by code line (if possible) and inspect all objects generated until you understand what they represent and how they fit together.

We truly hope that you find our book useful, whether you do population analysis for your research or for more applied goals, such as management or conservation biology. We even hope that you actually *enjoy* reading and working through it for its content, its style, and its presentation. In reality, we have written a book that we would have liked to have when we started our statistical population modeling in WinBUGS some years ago. If you have comments or find errors, please drop us an email at [marc.kery@vogelwarte.ch](mailto:marc.kery@vogelwarte.ch) or [michael.schaub@vogelwarte.ch](mailto:michael.schaub@vogelwarte.ch). We hope that WinBUGS frees the creative population modeler in you, as it has done for us.

*Marc and Michael,  
April 2011*

# Acknowledgments

---

We are indebted to three of our colleagues with whom we have collaborated for many years and who have directly or indirectly contributed much of the code, and more, documented in this book: Andy Royle, Olivier Gimenez, and Bob Dorazio. Over the years, they have been extremely generous in helping us to learn how to use WinBUGS efficiently and correctly. We thank the following people who have read and commented on parts or the book or helped otherwise: Andy Royle, Fitsum Abadi, Raphaël Arlettaz, Florent Bled, Richard Chandler, David Fletcher, Beth Gardner, Olivier Gimenez, Vidar Grøtan, Jérôme Guélat, Ali Johnston, Fränzi Korner Nievergelt, Bill Link, Mike Meredith, Jim Nichols, Marco Perrig, Tobias Roth, Beni Schmidt, and Giacomo Tavecchia. We are grateful to Steven Beissinger for writing an inspiring foreword. We furthermore thank the people who provided data sets, as well as the photographers who gave us their great shots of some of the organisms behind the numbers we crunch. The participants at our workshops (Sempach 2010 and 2011; Patuxent 2010) have been extremely important to try out what works and what does not and for honing our book, which is meant to be a gentle introduction to Bayesian statistical population modeling for exactly this kind of audience. Specifically, we are indebted to Andy Royle and his colleagues at Patuxent for hosting the BPA workshop in November 2011. We also thank our employers, the Swiss Ornithological Institute ([www.vogelwarte.ch](http://www.vogelwarte.ch)) and the Laboratory of Conservation Biology ([www.cb.iee.unibe.ch](http://www.cb.iee.unibe.ch)) at the University of Berne, for giving us creative time for research and writing. Finally, we feel a deep gratitude to our families, especially our wives Susana and Christine, for their love and patience and for granting us the freedom required to write this book.

# Introduction

## OUTLINE

1.1	Ecology: The Study of Distribution and Abundance and of the Mechanisms Driving Their Change	1
1.2	Genesis of Ecological Observations	6
1.3	The Binomial Distribution as a Canonical Description of the Observation Process	9
1.4	Structure and Overview of the Contents of this Book	13
1.5	Benefits of Analyzing Simulated Data Sets: An Example of Bias and Precision	16
1.6	Summary and Outlook	20
1.7	Exercises	21

### 1.1 ECOLOGY: THE STUDY OF DISTRIBUTION AND ABUNDANCE AND OF THE MECHANISMS DRIVING THEIR CHANGE

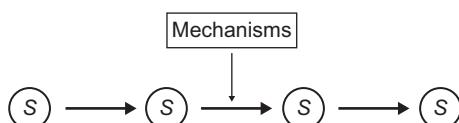
Ecology is concerned with the number (abundance,  $N$ ) of living things—how many individuals there are and how their number evolves over time, where they are and where they go to. Important questions concern their interactions with the abiotic and biotic environment, including each other, and what mechanisms drive these numbers and their dynamics. This classic view of ecology is reflected by the titles of two seminal textbooks: *The Distribution and Abundance of Animals* (Andrewartha and Birch,

1954) and *Ecology: The Experimental Analysis of Distribution and Abundance* (Krebs, 2001).

More generally, ecology can be described as the science that studies how states of biological systems interact with their environment and how this results in the temporal dynamics and spatial patterns of organisms that we observe. Figure 1.1 shows how state  $S$  evolves over time. The arrows connecting states between successive time periods denote the rate parameters that govern changes of state. State  $S$  may denote an individual state such as “alive” or the state of a collection of individuals, that is, population, such as “occurrence” or “local abundance,  $N$ ”. For the individual state “alive”, the arrows may represent the coin-flip-like survival process. For the abundance state ( $N$ ), the arrows may represent the demographic rates of survival, fecundity, immigration, and emigration. It is those rates on which the ecological mechanisms act to determine how a population is distributed in space or evolves over time.

A pervasive theme in ecology is that of hierarchical scales of organization—genes are nested within individuals, individuals within populations, populations within metapopulations or communities, and communities within metacommunities. Interestingly, this view of ecology is again reflected by the title of an influential ecology textbook: *Ecology: Individuals, Populations, and Communities* (Begon et al., 1986). These scales have biologically quite different meanings, and the practitioners of the associated branches of ecology often have very little in common with one another. And yet, it is fascinating to recognize that we can move among these scales simply by a redefinition of counted units (i.e., what we call an “individual”) and that they can be characterized by what is essentially the same set of quantitative demographic descriptors (Table 1.1).

At Scale 1, the unit is the classical individual living in a population (Table 1.1). It can move between states such as “alive” and “dead” or “newly recruited” and “not newly recruited”, thereby defining demographic rates such as survival and recruitment, respectively. Scale 1 represents the classic population concept. The interest is usually in understanding how biotic and abiotic factors impact vital rates (e.g., Newton, 1998) and



**FIGURE 1.1** The classic view of ecology as the science dealing with how the state  $S$  (for instance, local population size  $N$ ) of a living system evolves over time. These changes are governed by rate parameters (e.g., survival, fecundity, and dispersal in case of the abundance state). The interactions between the living system and its environment represent the ecological mechanisms that create the temporal dynamics we observe. An analogous scheme could be drawn also for spatial patterns.

**TABLE 1.1** Four Ecological Scales of Organization; the First Three of which Are Dealt with in This Book (see Royle and Dorazio, 2008, for the Fourth)

Scale of Organization = Type of Population	Description	
	Static (State Variable)	Dynamic (Vital Rates)
(1) One Site, One Species: Classic Population = Population of Individuals	Abundance $N$	Survival Probability ( $\phi$ ) Recruitment Rate ( $\gamma$ )
(2) Multiple Sites, One Species: Metapopulation = Population of (Local) Populations	$N_s$ $z_s$ Occupancy $\psi = \Pr(N > 0)$	Extinction Probability ( $1 - \phi_s$ ) Colonization Rate ( $\gamma_s$ ) Dispersal Rates
(3) One Site, Multiple Species: Community = Population of Species	$N_k$ $z_k$ Species Richness	Extinction Probability ( $1 - \phi_k$ ) Colonization Rate ( $\gamma_k$ ),
(4) Multiple Sites, Multiple Species: Metacommunity = Population of Communities	$N_{k,s}$ $z_{k,s}$ Species Richness	Extinction Probability ( $1 - \phi_{k,s}$ ) Colonization Rate ( $\gamma_{k,s}$ ) Dispersal Rates

*Notes:* All four can be represented as “populations” by simply redefining what represents an “individual” making up that population. These scales are hierarchical in the sense that each lower scale is included in a higher one. This means, for instance, that the quantitative descriptors of a classic population (Scale 1) may also be applied to the components of a metapopulation (Scale 2).  $N$ , abundance;  $z$ , presence/absence indicator;  $k$ , index for species;  $s$ , index for site.

how changes in vital rates translate into changes in numbers, that is, of population size (e.g., Sibly and Hone, 2002). Moving up one level, but still considering the individual unit, we have a collection of sites in which individuals can live. The movement probability among the associated populations (dispersal) is now an additional vital rate. The state variable is the size of the different populations.

At Scale 2, we view a single local population (or more generally, an occupied spatial unit) among a collection of potentially occupied spatial units as the item, and thereby obtain a metapopulation (Hanski, 1998). The basic, static descriptor of a metapopulation is the set of  $N_s$  values, that is, classic abundance at each spatial unit  $s$ . A less information rich, yet easier to measure version is the occupancy state  $z = I(N > 0)$ , where  $I()$  denotes the indicator function that evaluates to 1 for an occupied unit and zero for an unoccupied one. The population average of  $z_s$  is called “incidence” in the metapopulation literature (e.g., Hanski, 1994, 1998) or occupancy probability,  $\psi$  (e.g., MacKenzie, 2006). Occupancy and abundance are directly related to each other via  $\psi = \Pr(N_s > 0)$ , that is, occupancy probability is simply the probability that abundance at a site is greater than zero (Royle and Nichols, 2003). So, clearly, there is a sense in which “distribution and abundance”

in the book titles cited above is redundant; the characterization of a metapopulation by local abundance is fully sufficient and directly yields a description in terms of occupancy (Royle et al., 2005, 2007b).

Metapopulation ecology has been a part of ecology's mainstream for several decades now (Levins, 1969; Hanski, 1994, 1998; Hanski and Gaggiotti, 2004) and has been extremely influential in conservation biology, for instance, by highlighting the importance of random extinctions of local populations even at sites with suitable habitat, and consequently, by stressing the importance of connectivity among subpopulations as a means of avoiding permanent extinction of patches. In a similar vein, metapopulation biology provides the understanding for why currently unoccupied habitat patches may be as important for the long-term survival of a species as currently occupied ones (Talley et al., 2007). The dynamic descriptors of a metapopulation are analogous to those of a classic population, except that "individuals" (=occupied sites, local populations) can be reborn, that is, go extinct and yet later the site may be recolonized. Metapopulation-like dynamic models of occurrence proved insightful in epidemiology and disease ecology and have been used to model the spread of a disease (e.g., West Nile virus, Marra et al., 2004) or invasive species (Wikle, 2003; Hooten et al., 2007; Bled et al., 2011b).

An alternative way to quantify the total occurrence of an organism in some area is simply the sum of occurrences (i.e.,  $\sum_s z_s$ ); this represents a "population size" of occupied spatial units. Both the ratio  $\psi$  and the sum of  $z_s$  characterize the *range* or *distribution* of an organism. Ranges are the focus of macroecology and biogeography (Brown and Maurer, 1989; Gaston and Blackburn, 2000). Many ecological studies aim to predict species occurrence (i.e.,  $z_s$ ) from habitat or other local site attributes (e.g., Scott et al., 2002), either for fundamental reasons, for example, to study a species' niche (Guisan and Thuiller, 2005), or for applied reasons, for example, to predict the location of previously undetected occurrences, or to determine the most suitable sites for reintroduction projects. In essence, these models focus on the extent of a metapopulation, and the latest of them try to incorporate biological interactions (such as the possibility for an unoccupied site to become recolonized from an occupied site nearby; Guisan and Thuiller, 2005), thus bringing them increasingly closer to a classical and more mechanistic, metapopulation model of a species distribution.

Another increasingly common example of an occupancy study is a distribution atlas (Hagemeijer and Blair, 1998; Schmid et al., 1998; see review in Gibbons et al., 2007) that documents distribution ranges, for instance, by the presence or absence of a species in each cell of a grid. The data collected during such atlas studies, when repeated over time in the same area, have become an important raw material for studies documenting effects of climate change on species ranges (Thomas and Lennon, 1999; Huntley et al., 2007). Finally, occupancy is an important state variable for biodiversity monitoring,

for example, in the Swiss biodiversity monitoring program BDM (Weber et al., 2004, also see [www.biodiversitymonitoring.ch](http://www.biodiversitymonitoring.ch)), in amphibian monitoring (Pellet and Schmidt, 2005), and as one of the important and most widely used criteria by which the IUCN Red list status of a species is assessed ([www.iucnredlist.org/about/red-list-overview#redlist\\_criteria](http://www.iucnredlist.org/about/red-list-overview#redlist_criteria)).

Moving up another level among the ecological scales of organization, a community can be conceived of as a “population” of species at a single site (Table 1.1, Scale 3). A community can be described at a point in time by the species–abundance distribution,  $N_k$  (Engen et al., 2008). A simpler community description is the sum of individual species’ occurrences, that is, species richness ( $\sum_k z_k$ ). Species richness and its dynamic components are the central focus of research in many branches of ecology such as biogeography (Jetz and Rahbek, 2002), as well as conservation science, for instance, when looking for hotspots of species richness to direct conservation funds (Orme et al., 2005). Indeed, species richness is the most widely used measure of biodiversity (Purvis and Hector, 2000) and is frequently used in monitoring programs (e.g., Weber et al., 2004; Pearman and Weber, 2007).

At the highest level of ecological scales of organization (Table 1.1, Scale 4), a metacommunity is a set of population of multiple species at many sites. Metacommunities have recently taken center stage in ecology with the neutral theory of biodiversity (Hubbell, 2001; Gotelli and McGill, 2006). In terms of its quantitative description, a metacommunity can be dealt with fairly analogously to a community (Table 1.1).

Of course, not every ecologist focuses directly on the population descriptors of Table 1.1. For instance, evolutionary, behavioral, or physiological ecologists deal with the interactions among individuals and with the environment that may become the mechanisms determining the size ( $N$ ) and dynamics of a population (Sibly and Calow, 1986; Stearns, 1992; Krebs and Davies, 1993; Sutherland and Dolman, 1994). However,  $N$  remains important implicitly: because in order to be ecologically relevant, any evolutionary, behavioral, or physiological mechanism must ultimately have at least the potential to affect  $N$ .

The modeling of these hierarchical scales may be conducted very naturally in a hierarchical manner, that is, a metapopulation can be modeled in terms of patch occupancy  $z_s$  or in terms of the local population size  $N_s$ . Similarly, its dynamics can be expressed by the survival and colonization probabilities of patches or by the survival and recruitment probabilities of the individuals occupying these patches and by their dispersal among the patches. Analogous alternative descriptions in terms of the state and the dynamics are possible for communities and metacommunities. One important descriptor of the dynamics of all four levels is the sustained rate of change of the system, or trend. Trend is a consequence of survival, recruitment, and dispersal probabilities and thus a derived quantity rather

than a driver of the system. Nevertheless, it is the simplest and most parsimonious description of system dynamics and of tremendous practical importance in many applications of population ecology, such as conservation biology and wildlife management (Balmford et al., 2003).

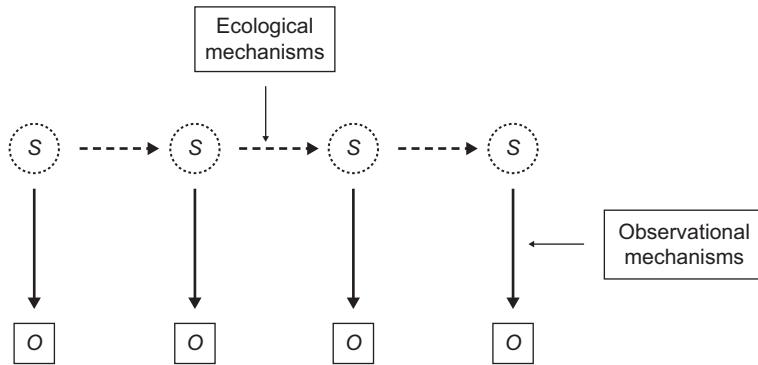
In summary, the three key state variables used to describe populations, metapopulations, communities, and metacommunities are abundance, occurrence (distribution), and species richness (Royle and Dorazio, 2008). From a pure modeling point of view, all three simply represent variants of a “population” that can be described by its size. In addition, there are the parameters that govern the dynamics of these state variables: survival/extinction, fecundity, colonization, and dispersal (immigration and emigration). Collectively, we call the study of these demographic quantities *population analysis*. Population analysis permeates a large part of ecology and of its applications such as conservation biology or fisheries and wildlife management. Indeed, it could be argued that population ecology, which we see as somewhat synonymous with population analysis, is a central pillar of the entire discipline of ecology.

## 1.2 GENESIS OF ECOLOGICAL OBSERVATIONS

A widely ignored consideration regarding all these varieties of the state variable, along with their dynamic rates, is that they are usually not directly observable; rather, “individuals” of all kinds can be overlooked; their *detection probability* is not perfect (i.e.,  $p < 1$ ; Schmidt, 2005). Therefore, a more accurate view of ecology is depicted in Fig. 1.2. This *hierarchical* view considers all observations in ecology as a result of two coupled processes: an ecological process, which usually is the focus of our interest, and an observation process, which is conditional on (i.e., whose result depends on) the result of the ecological process (Royle and Dorazio, 2006, 2008). In most ecological studies, the state of the system and its dynamics are latent. Therefore, they must be inferred from the observations  $O$  by modeling the main features of the observation process.

The ecological process itself is influenced by mechanisms that may be deterministic (e.g., habitat) or stochastic (e.g., demographic or environmental stochasticity) and that together determine the state of a system, for example, the population size,  $N$ . However, our observations of the system are also the result of an observation process, which may again be influenced by a variety of factors, among them deterministic (e.g., habitat-dependent observation errors) and stochastic mechanisms. Our observations in ecology are thus always a combination resulting from ecological and observation mechanisms.

For instance, assume that more birds are counted (i.e., *observed*) in habitat A than B. This can mean that there really *are* more birds in A than B, but



**FIGURE 1.2** A hierarchical view of ecology that acknowledges the fact that state  $S$  is fully or partly unobservable (latent). Ecological mechanisms (representing the ecological process) affect the dynamics of the latent state, while observational mechanisms constitute the observation process. The observation process maps the latent state  $S$  (for instance, local population size  $N$ ) on the observation  $O$  (for instance, a count). To emphasize the latent nature of the state process, arrows connecting states and circles around  $S$  are dashed.

it can also mean that birds are simply more *visible* in A than B or indeed any combination of the two. Similarly, not only the mean but also the variability of the observations is made up of these two components: one coming from the ecological process and the other from the observation process (e.g., nondetection error, sampling error; Royle and Dorazio, 2008, pp. 11–13). In most branches of ecology, we are thus faced with a situation where we have incomplete knowledge about an ecological system under study, and we must use error-prone observations to infer its characteristics, such as state variables or dynamic rates and the kind and strength of their interactions with the environment. In short, in the study of ecological systems, we must account for the fact that detection probability ( $p$ ) of all three kinds of “individuals” shown in Table 1.1 is usually less than 1.

Direct inference based on the raw observations in ecology, and disregard of the observation process, may be risky. If nondetection error (rather than, say, false positives or double counting) represents the main feature of the observation process (i.e.,  $p < 1$ ), population size, distribution, or species richness will all be underestimated (Schmidt, 2005). Similarly, estimates of dynamic population descriptors will be biased. For instance, survival probabilities will be underestimated (Nichols and Pollock, 1983; Martin et al., 1995; Gimenez et al., 2008); extinction and turnover rates in metapopulations, communities, and metacommunities will be overestimated (Nichols et al., 1998b; Moilanen, 2002); and the perceived strength of a relationship between survival, abundance, or occurrence and environmental covariates will be underestimated (Tyre et al., 2003; MacKenzie et al., 2006); also see Section 13.2. It has not been sufficiently widely

recognized that what is typically called a distribution map in ecology (see, e.g., Scott et al., 2002), may in fact simply be a map of the difficulty with which an organism is found (Kéry et al., 2010; Kéry, 2011b). For instance, any spatially varying mechanism such as local density that causes a species to be more likely to be detected at some sites than at others will leave its imprint on a map of putative species distribution.

When imperfect detection is not accounted for in the modeling of ecological systems, the observed variation in the system (e.g., variance in population size) will typically be greater than the true variation in the ecological system (e.g., Link and Nichols, 1994). Some sort of variance decomposition must then be employed to separate true system variability from variability that is due to observation error (Franklin et al., 2000; De Valpine and Hastings, 2002). Such a partitioning of the observed variance is particularly important for population viability analyses (Lindley, 2003), investigations of density dependence (Dennis et al., 2006; Lebreton, 2009), and the setting of harvest regulations (Williams et al., 2002).

Consequently, we think that it is important in population analysis to include the essential features of the observation process when making inferences from imperfect observations about the underlying ecological process, for example, about the quantitative descriptors of all ecological scales of organization depicted in [Table 1.1](#). Otherwise, we risk describing features of the observation process rather than of the ecological process we are really interested in. We need special data collection designs and methods of interpretation of the resulting data (i.e., models) that take explicitly into account the observation process to tease apart the genuine patterns in the ecological states from those induced in the observations by the observation process (MacKenzie et al., 2006). That is, when the quantities in [Table 1.1](#) need to be studied directly, they must be *estimated* from, and cannot usually be equated with, the observed data.

To explicitly accommodate both the ecological and the observation process, an emerging and very powerful paradigm for population analysis is that of hierarchical models (Link and Sauer, 2002; Royle and Dorazio, 2006, 2008), sometimes also called state-space models (Buckland et al., 2004). One reason why these models are so useful for population analysis is that they simply replicate the hierarchical genesis of ecological data on animals and plants depicted in [Fig. 1.2](#): one level in the model is the un- or only partially observed true latent state (e.g., being alive, abundance, or occurrence) and another level is the observation process, typically represented by detection probability  $p$ . Among the several advantages of hierarchical models is that they achieve a clear segregation of the observations into their two (or more) components. Thus, these models greatly foster intellectual clarity.

In this book, we follow Royle and Dorazio (2008) in emphasizing the distinction between the true state of an individual, a population, or

a community, and their observed state, and that the two are linked by an observation process, which imperfectly maps the former onto the latter. An explicit modeling of the observation process is thus essential to our approach of population analysis. Because we are convinced that most ecologists learn best by seeing examples, we next provide a brief numerical illustration for the observation process that shows why it is so important to consider it when making an inference in population ecology.

### **1.3 THE BINOMIAL DISTRIBUTION AS A CANONICAL DESCRIPTION OF THE OBSERVATION PROCESS**

To better understand the key features of the observation process behind most ecological field observations, let us assume 16 sparrows live in our yard and that their population size was constant over a few weeks during which we make some observations (i.e., count them) to find out how many there are. Let us assume that there are no false-positive errors, only false-negative errors. This means that one sparrow cannot be counted for another and that another species cannot erroneously be identified as a sparrow. Let us further assume that each sparrow is independently observed or heard with a constant detection probability of 0.4. This means that if we step out into our yard 10 times, we will expect to see or hear (i.e., detect) that particular sparrow about 4 times. Of course, these are all abstractions of the real-world observation process, but they are very often plausible and adequate assumptions.

When we are interested in the total count of sparrows in our yard, then we have just defined a binomial random variable with sample or trial size  $N = 16$  and so-called success probability  $p = 0.4$ . The binomial distribution is the mathematical abstraction of situations akin to coin flipping, where an event can either happen with a certain probability ( $p$ ) or not (with  $1 - p$ ), and we watch a number of times ( $N$ ), all assumed independent, and count how many times ( $C$ ) that event happens. The event here is the detection of an individual sparrow, and  $N$  is the latent state of the local population size of sparrows in the yard. Since detection is a chance process, we will typically not wind up with the same count all the time when we repeat the exercise. We can use a physical model, for example, the flipping of one or several coins, or a computer model to study the features of the observation process.

As we will see throughout the book, program R (R Development Core Team, 2004) is great for conducting quick and simple simulations to better understand a system. We first define the constants in the system:

```
N <- 16      # Population size of sparrows in the yard  
p <- 0.4    # Individual detection probability
```

To simulate a single count, we simply draw a binomial random number with sample size  $N$  and success probability  $p$  (of course, most of you will get different simulated counts from those shown here).

```
rbinom(n = 1, size = N, prob = p)
[1] 11
```

So the first time, we count 11 sparrows. The next time we go out into the yard, we count again:

```
rbinom(n = 1, size = N, prob = p)
[1] 4
```

Now only four! This is a large difference to the previous count, so we count again a little later.

```
rbinom(n = 1, size = N, prob = p)
[1] 9
```

This count is more similar to the first one. But perhaps we are a little worried about the variability in these counts. Were the lower counts made under somewhat inferior conditions? Or did we not pay so much attention to the counting then?

We can very simply use R to study the long-term behavior of the assumed observation process in our example: we simply draw a large number  $n$  of samples from the binomial random variable defined by  $N$  and  $p$  and summarize that sample. Let us draw one million then, since this does not cost us anything and R is free.

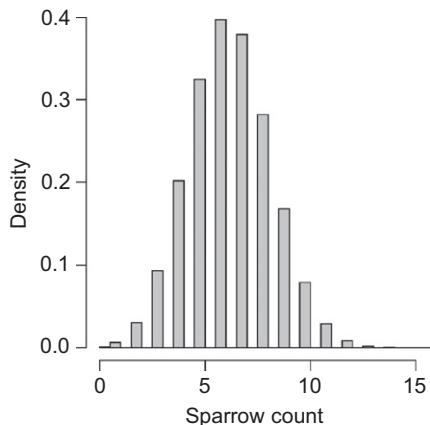
```
C <- rbinom(n = 10^6, size = N, prob = p)
```

Next, we describe that big sample in a graph (Fig. 1.3) and numerically to better understand some of the essential features of the counting process, that is, of the observation process behind our counts of a sparrow population of size 16.

```
mean(C)
[1] 6.404259
var(C)
[1] 3.842064
sd(C)
[1] 1.960118
hist(C, breaks = 50, col = "gray", main = "", xlab = "Sparrow count",
     las = 1, freq = FALSE)
```

This simple example illustrates several features about an observation process that is dominated by nondetection error (i.e., where misclassification and double counts are absent):

1. The typical count  $C$  is smaller than the actual population size  $N$ . Indeed, the mean of a binomial random variable and hence the expected count



**FIGURE 1.3** Frequency distribution of the observations, when 16 sparrows are counted repeatedly and each has independently a detectability of 0.4. The binomial distribution is our canonical model for the observation process involving detections of plant or animal “individuals”, where “individual” can mean many different things (see Table 1.1).

of sparrows, equals the product of  $N$  and  $p$ . This should be 6.4 in the sparrow example and in our sample we get pretty close to that.

2. However, the counts vary quite a lot, *even under totally identical conditions*. Indeed, some of the counts in our sample were 0 and one was 16, meaning that sometimes not a single sparrow was detected but another time, all 16 of them were. Hence, there is nothing intrinsically wrong or inferior with smaller counts. Smaller counts may simply result from the random nature of the counting process in the presence of imperfect detection. Thus, any count with  $p < 1$  will automatically tend to vary from trial to trial. Unless  $p = 0$  or  $p = 1$ , it is impossible to eliminate that variation by the sampling design or standardization (though other components of variation may be eliminated).
3. Actually, not only the mean count but also the magnitude of the variation of counts is known from statistical theory. The variance is equal to the product of  $N$ ,  $p$ , and  $1 - p$  and thus should be around 3.84. Up to sampling variation, the observed variance of the 1 million counts of sparrows is identical to that.

False negatives will not only affect population counts and thus estimates of population size but also parameter estimates derived from the counts, such as survival or state-transition probabilities. In addition, the observation process will often be affected by explanatory variables and perhaps even by the exact same ones in which we are interested for the ecological process (see Figures 12.3. and 13.2.). Unless detection probability  $p$  is estimated then, any patterns in  $p$  will be perceived in the apparent state of the

ecological process. For instance, one can often read that state-space models (Chapter 5) correct for observation error. In truth, they only do so in a rather vague way. They only account for the binomial sampling variation around a mean count (i.e.,  $Np$ ), but cannot correct for the general bias in the counts relative to true population size, nor any patterns (e.g., over time) induced in counts by patterns in  $p$  (see Section 5.3). These latter two kinds of observation error (detection bias and patterns in detection) cannot be corrected for by the methods in Chapter 5 unless one has extra information about the detection process and uses the methods in Chapters 6, 10, 12, and 13.

We have claimed that the binomial distribution is the canonical description of the observation process. This is true in the sense that it underlies the vast majority of statistical methods that correct for imperfect detection in population analysis (Buckland et al., 2001; Borchers et al., 2002; Williams et al., 2002; Royle and Dorazio, 2008). However, other statistical distributions may be adopted as a description of the observation process in some cases, for instance, a beta-binomial distribution to account for nonindependent detections (Martin et al., 2011). The Poisson distribution is an appropriate description of the observation process if encounter frequencies rather than simply detection events are observed, for example, when we know that  $y_1$  animals were observed once,  $y_2$  twice, and so forth. In fact, there is a natural relationship between a Poisson and a binomial model of detection because the binomial detection probability may be expressed as the probability of observing Poisson counts greater or equal to one (Royle and Gardner, 2011). Finally, the negative binomial distribution may be adopted to model over-dispersed encounter frequencies, such as arise from nonindependent detections of individuals, for instance, animals in groups (Boyce et al., 2001).

The binomial distribution is a useful starting point for modeling an observation process that is dominated by false-negative errors, that is, where false-positive errors are rare or absent. False-positive errors arise when one “individual” in Table 1.1 is mistakenly identified as another one. Variants of this kind of error include unoccupied sites being identified as occupied when modeling species distributions or mistakenly identifying one species for another when modeling communities.

In the fields of statistics covered in this book, almost all methodological development over the last decades has dealt with false-negative errors; false-positive errors were essentially assumed away. There are two reasons for this. First, false-negative errors are ubiquitous in ecology and probably much more widespread than false positives. Moreover, false positives can more easily be eliminated by the design of a study, by good training of a field crew or adequate calibration of a measuring device. For instance, any doubtful records, for example, detections of a species when estimating the size of a community (see Section 6.3) may simply be eliminated (J. D. Nichols, personal communication). This will weed out false positives at the risk of losing some valid detections of species. The latter will simply

lower detection probability but obviously not bias estimators from models that account for false-negative errors.

However, a second reason for why false positives have received much less attention than false negatives is that they are harder to deal with mathematically. Although on the whole, false positives may have less biasing effects in population analyses than false negatives, they are nevertheless important to account for in some situations. It has been shown that even relatively rare false-positive events may induce strong biases in site-occupancy models (Royle and Link, 2006). Interestingly, the effects of false positives often become stronger with larger sample sizes, for instance, when more surveys are conducted at each site in the context of site-occupancy sampling (see Chapter 13). Thus, it is important to watch out for this kind of error and if necessary account for it in population analyses. Therefore, it is encouraging that during the last decades a steadily increasing number of papers has dealt with false-positive errors, including Kendall et al. (2003), Nichols et al. (2004), Lukacs and Burnham (2005), Pradel (2005), Royle and Link (2006), Nichols et al. (2007), Conn and Cooch (2009), MacKenzie et al. (2009), Wright et al. (2009), Yoshizaki et al. (2009), Link et al. (2010), McClintock et al. (2010), and Miller et al. (2011). It is likely that we will see much more developments in this important topic in the near future.

In summary, we believe that it is important to account for the observation process when making an inference about any quantity in [Table 1.1](#) for populations of animals or plants in population ecology, management, and conservation.

## 1.4 STRUCTURE AND OVERVIEW OF THE CONTENTS OF THIS BOOK

In this book, we deal with populations of the first three kinds in [Table 1.1](#) and most intensively with classical population size and the demographic processes underlying its change (survival, fecundity, and dispersal or movement) and with the “population size” of occupied patches, that is, species distributions or occupancy, and its dynamic rates. We also see a little bit of species richness, that is, the size of the third kind of “population” (Section 6.3). The structure of this book is summarized in [Table 1.2](#). We present this overview by distinguishing population studies at single sites from those at multiple sites, which we call studies with a metapopulation design (Royle, 2004c; Kéry and Royle, 2010). The scheme in [Table 1.2](#) covers a fairly large part of the questions and models that ecologists might want to consider when analyzing populations of plants and animals.

We start with simple models for counts in Chapter 3 and then introduce more complexity via random effects in Chapters 4 and 5; it is the presence of

**TABLE 1.2** Structure of This Book and Loose Chapter Overview

Quantity Modeled	Single Site	Multiple Sites
<b>Distribution</b>		
Apparent Distribution	–	Logistic Regression (Chapter 3)
True Distribution	–	Site-Occupancy Model (Chapter 13)
<b>Abundance</b>		
Apparent Abundance	Poisson GLM (Chapter 3) Poisson GLMM (Chapter 4) State-space Model (Chapter 5) Integrated Population Model (Chapter 11)	
True Abundance	Closed Population Capture– recapture Model (Chapter 6) Jolly–Seber Model (Chapter 10) Integrated Population Model (Chapter 11)	Binomial mix Model (Chapter 12)
<b>Vital Rates</b>		
Survival Probability	Cormack–Jolly–Seber model (Chapter 7) Ring-recovery Model (Chapter 8) Multistate Model (Chapter 9) Jolly–Seber Model (Chapter 10) Integrated Population Model (Chapter 11)	
Fecundity/Recruitment	Poisson GLM (Chapter 3) Jolly–Seber Model (Chapter 10) Multistate Model (Chapter 9) Integrated Population Model (Chapter 11)	
Movement Probability	–	Multistate Model (Chapter 9)
Leslie-Matrix Modeling	Integrated Population Model (Chapter 11)	

*Notes:* A distinction is made between models that can be applied to data from a single study site and those that require data from multiple sites. All former can also be applied to data from multiple sites.

these random effects, which converts the models in Chapter 3 to hierarchical models. The so-called state-space models of Chapter 5 attempt a partitioning of the total variation in the observations into one portion due to the ecological system and another portion due to observation error. We have claimed previously that the only accounting possible in this kind of model is for the

random sampling variation (due to imperfect detection) in the counts, but not for detection probability proper. Hence, all models in Chapters 3–5 deal with apparent, not true abundance, where “true” to us means “corrected for imperfect detection”. In Chapter 6, we encounter for the first-time models that achieve a comprehensive accounting for detection error: these are the classical capture–recapture models for closed populations.

In Chapters 7–11, we move on to the quantities driving the changes in population size: demographic or vital rates. We note in passing that vital rates that represent probabilities such as survival are still often called rates, but this is not completely correct: a rate, unlike a probability, is not bounded by 0 and 1. We model survival probabilities based on two kinds of data, resighting and dead recoveries, respectively, as well as movement probabilities between states. These models again fully account for detection error. As an aside, in Chapter 3, we show a simple example of the modeling of another important dynamic rate: fecundity. Chapter 10 introduces the Jolly–Seber model, where inferences about abundance, survival, and recruitment from capture–recapture data can be made. In Chapter 11, we introduce a sort of synthesis of several models in preceding chapters and fit simple population models of the Leslie-matrix type. One important feature of these integrated population models is that by the joint modeling of several data sets and data types, parameters can often be estimated that are not identifiable with each data set alone.

Most methods for population analysis at single sites (in the left column in the body of [Table 1.2](#)) can also be used for populations at multiple sites. In the last two main chapters of the book, we present two classes of models that explicitly focus on distribution and abundance in a metapopulation setting: the binomial (or N-)mixture model for abundance (Chapter 12) and the site-occupancy model for species distributions (Chapter 13).

There are several important, recurring themes throughout our book. The three most essential ones are actually recurrent themes in much of applied statistical modeling:

1. Linear models
2. Generalized linear models
3. Random effects, or more generally, hierarchical modeling

First, in linear models, the mean of a response is thought to be made up of additive effects of covariates. This is one of the most widespread manner in which relationships between a response and explanatory variables are modeled. If you understand how to build a linear model using a design matrix (Chapter 3), you will achieve an extraordinary flexibility in your modeling. Second, generalized linear models (GLMs) use the same concept of linear models and the design matrix and carry that over to nonnormal responses, such as Poisson or binomial random variables. A Poisson or binomial response may also be the components in a

larger model, for instance, to model survival events over a number of years. We may then encounter a GLM as part of a larger model. We will meet GLMs all over in this book and give a concise introduction to them in Chapter 3. Finally, random effects are another key concept in applied statistical modeling and are introduced in Chapter 4. They give considerable flexibility to our modeling of variation and of correlations. Like GLMs, random effects, or hierarchical models, permeate this entire book. A GLM containing random effects is typically called a generalized linear mixed model (GLMM). We believe that if you understand linear models, GLMs, and random effects, then you understand a large part of applied statistical modeling. You then achieve an organic understanding of many of your models and will be able to build your custom models in a modular, creative, and efficient way (Kéry, 2010). In particular, you may then start to see hierarchical models as a natural way of describing complex stochastic systems by a nested sequence of component GLMs.

Other important concepts or techniques that you may be particularly interested in include the following:

- The distinction between an implicit and an explicit hierarchical model (Chapters 4–6, also see Royle and Dorazio, 2008)
- Data augmentation (Chapters 6 and 10)
- Posterior predictive checking of model adequacy (Chapters 7 and 12)
- The assessment of parameter estimability (Chapter 7)

We emphasize that by combining these basic concepts in a creative way, you will be able to create a large variety of statistical models that will help to obtain better inference from your data.

You will see plenty of WinBUGS and R code in this book. We use Courier font to highlight code. We comment our code quite extensively to make it easier to understand. In both R and WinBUGS, comments are flagged with a hash (#) sign, which means that the line following the hash is ignored.

## **1.5 BENEFITS OF ANALYZING SIMULATED DATA SETS: AN EXAMPLE OF BIAS AND PRECISION**

---

One key feature of this book is that we work a lot with simulated data sets. There are tremendous benefits in doing so. As argued elsewhere (Kéry, 2010), the advantages of simulating (=assembling) data sets and then analyzing (=disassembling or breaking apart) them again are manifold:

1. Truth is known, so we know what to expect from the analysis.
2. We get a check for whether we have coded things correctly.
3. We can experience sampling error, that is, the variation in results. For instance, we may study long-run characteristics of estimators such as bias or precision (see example below).

4. In particular, repeatedly simulating data under some conditions and analyzing them provides an extremely flexible way of conducting power analyses.
5. We can check the effects of assumption violations: simulate data under a different model than that which is used to analyze a data set.
6. Finally, we can prove to ourselves that we understand an analysis: when you can assemble, that is, generate a data set under a model, you can also disassemble it, that is, break it down again in the analysis of that model.

As we will see many times in this book, we simulate data “from the inside out”, that is, we first decide on values for any covariates and on coefficients that relate these covariates to the mean response, and then we add residual variation by drawing random variables with a given expectation and possibly a specified variance. When we disassemble the data set, we go the opposite way and break apart the full response into its ingredients, that is, covariate effects, random effects, etc. Arguably, you will only be able to simulate data under a model if you have really understood that model!

As an illustration, let us look at the concepts of bias and of the precision of an estimator. Both are often misunderstood by ecologists. One can often read an *unbiased estimate*. Strictly, this is wrong because estimates cannot be unbiased, only *estimators* can (Link and Barker, 2010). Estimators are the procedures that produce a particular estimate and if the estimates they produce are right on target, on average, an estimator is said to be unbiased. Right on target means that the mean of all estimates an estimator produces is the same as the population quantity (the parameter) estimated by that estimator. This may sound like counting green peas, and perhaps in ecological writings one could accept *unbiased estimate* as synonymous with the more accurate *unbiased estimator*. However, we must be able to make a clear distinction between a population quantity (a parameter), an estimator (a procedure), and a particular estimate of the population quantity produced by that procedure. Each individual estimate can be far off the target (the parameter), meaning the precision of the estimator is low, and yet the average of all estimates produced by the same procedure may be right on the mark, meaning the estimator is unbiased. Thus, the precision of an estimator expresses the similarity of repeated estimates produced by an estimator. Bias and precision are different aspects of the quality of an estimator: an estimator can be biased but precise, unbiased but imprecise (MacKenzie et al., 2006), or any other combination of these terms.

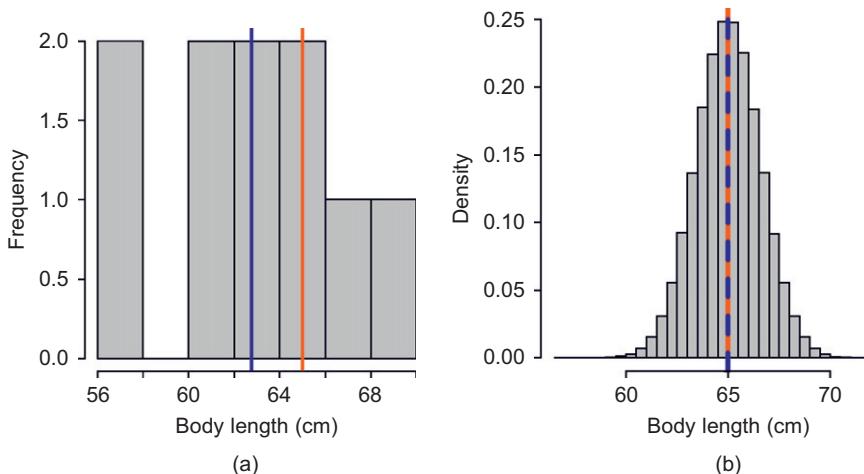
We next illustrate bias and precision by simulation: we simulate data from a large population, for which we want to estimate the population mean. We assume that we estimate body size of adult asp vipers (Fig. 1.4) by taking a sample of 10 snakes, measuring them, and calculating the mean



**FIGURE 1.4** Asp viper (*Vipera aspis*), Germany, 2009 (Photograph by T. Ott).

length. We will see that the mean from one individual sample of 10 will hardly ever be right on target; rather, due to sampling variation (individuals in the population differ and we only sample 10 of them), the mean of 10 will be lower or higher than the true population parameter. However, *repeatedly* sampling that population, by measuring the length of 10 snakes, and producing a histogram of these means of 10 will show that the sample mean is an unbiased estimator of the population mean. Moreover, from statistical theory we know by how much, on average, means of 10 vary around the true population mean: this is given by the standard error of the mean.

So here we go: we assume that full-grown asp vipers in the Jura mountains average 65 cm with a standard deviation of 5. We sample 10 snakes, measure their length and use that sample to say something about the population mean.



**FIGURE 1.5** Histogram of body length of 10 asp vipers (a) and histogram of 1 million sample means of the body length of 10 asp vipers (b). Red line: population mean, blue line: mean of sample (means).

```
# Population values for mean and standard deviation of individual
# lengths
mu <- 65           # Population mean
sigma <- 5          # Population SD

# Draw a single sample of 10 and summarize it
x <- rnorm(n = 10, mean = mu, sd = sigma)
```

Figure 1.5a shows one data set; do not forget that yours will be different. The red line indicates the population mean (i.e., the value of the parameter that we want to estimate when calculating the sample mean), and the blue line is the mean for our particular sample of 10 snakes. We see that here the sample mean is too small relative to the population mean.

To see whether the difference between the blue line and the red line is simply sampling variation, we can repeatedly draw such samples and plot the means. Hence, let us use R and draw 1 million samples of 10 snakes and plot their means (Fig. 1.5b).

```
reps <- 10^6
sample.means <- rep(NA, reps)
for (i in 1:reps) {
  sample.means[i] <- mean(rnorm(n = 10, mean = mu, sd = sigma))
}

# Produce figure
par(mfrow = c(1, 2), las = 1)
hist(x, col = "gray", main = "", xlab = "Body length (cm)", las = 1)
abline(v = mu, lwd = 3, col = "red")
abline(v = mean(x), lwd = 3, col = "blue")
```

```
hist(sample.means, col = "gray", main = "", xlab = "Body length (cm)",  
nclass = 50, freq = FALSE, las = 1)  
abline(v = mu, lwd = 5, col = "red")  
abline(v = mean(sample.means), lwd = 5, col = "blue", lty = 2)
```

We see that the distribution of the sample means in Fig. 1.5b is much more narrowly centered on the true population value. Moreover, from theory we know the spread of the distribution of sample means, that is, we know the precision of the mean as an estimator of the population mean: this is given by the standard error of the mean, that is, by  $sd(x)/\sqrt{n} = 5/\sqrt{10} = 1.58$ . Let us see how closely we get to that in our simulation. Remember, the standard error of our parameter estimator (here, the sample mean as an estimator of the population mean) is the standard deviation of the distribution of the estimates.

```
sd(sample.means)  
[1] 1.581443
```

This is pretty close. You can make your estimate of the standard error of the mean arbitrarily exact by increasing your simulation sample size ( $n$ ).

We always need to clearly distinguish between sample and population quantities. A parameter is a feature of the population and we want to estimate it based on some sample statistic (something that can be measured in the sample), such as the sample mean. Sample statistics, and therefore our parameter estimates, are affected by sampling variation: not all individuals are identical and we only measure a sample, not the entire population. Sampling variation may be quantified by the standard error of our estimator; this quantifies the precision of an estimator. The standard error of the sample mean as an estimator of the population mean can be estimated by the population standard deviation divided by square root of the sample size. Here, the standard error is the standard deviation of a hypothetical distribution of parameter estimates (means) from a sample size of 10. It may be confusing that the standard error is in fact a standard deviation, but one characterizing the variability in a collection of *parameter estimates*, rather than the variability among individual *measurements*.

## 1.6 SUMMARY AND OUTLOOK

Distribution and abundance, along with their dynamic rates of change such as survival or extinction probability, and the factors that affect them, lie at the heart of the science of ecology. Alas, almost universally, neither state nor rate parameters in natural populations of animals and plants can ever be observed without error. In particular, detection error (manifest in the presence of false-negative observations) is a hallmark of ecological observations of populations. We have seen that the binomial distribution is the typical description of an observation process that is dominated

by false-negative errors, but that other statistical distributions may sometimes be appropriate, for example, the beta-binomial, Poisson, and negative binomial. We have seen that the opposite type of error, false positives, has received far less attention in the literature so far, but that it is the focus of much recent work. We think that the observation process should be accounted for as much and as often as possible in the analysis of wild populations. Therefore, much of this book covers methods that attempt a clean partitioning of the ecological and the observation processes that underlie ecological observations. This partitioning is often achieved using hierarchical models, where separate model components describe the latent ecological process and the observation process. We are big fans of analyzing simulated data sets; perhaps the single biggest advantage of using simulated data sets is that the mere act of simulating data enforces an almost complete understanding of a model. Next, after having motivated the population analysis part in the book title, we give a very brief review of the other half of our book's title, namely "Bayesian" and "WinBUGS".

## 1.7 EXERCISES

---

1. Detection probability: convince yourself that very few quantities in nature are ever perfectly detectable. Stand at the window for 1 min and make a list of all the bird species that you see. Repeat this once or twice, then compare the lists among times and observers. If detection were really perfect (for all species, at all times, for all observers, etc.), then all the lists would be the same for all observers and it would not matter for how long you watched. Essentially, you would detect all species instantaneously.

You may conduct that exercise with a quantity of your choice: for example, the number (or identity) of people in your office hall, and the number of people in your bus. Alternatively, you could also count the brands of cars that pass in front of you.

2. Intrinsic variability of counts: it is our experience that ecologists often do not realize that counts vary intrinsically as soon as detection is imperfect. Moreover, counts that vary more are sometimes viewed as being of inferior quality than counts that vary less, or, that the observers producing these counts are better or worse. It is true that, everything else equal, sloppy counts will usually be more variable than those made by a more dedicated observer. However, owing to the mean–variance relationship in binomial counts, two of the most important factors affecting the variability of counts is (1) the number of things available for counting (i.e.,  $N$ ) and (2) detection probability,  $p$ . Produce a plot or a table that makes you better understand these relationships.

# Brief Introduction to Bayesian Statistical Modeling

## OUTLINE

2.1	Introduction	23
2.2	Role of Models in Science	24
2.3	Statistical Models	27
2.4	Frequentist and Bayesian Analysis of Statistical Models	28
2.5	Bayesian Computation	38
2.6	WinBUGS	38
2.7	Advantages and Disadvantages of Bayesian Analyses by Posterior Sampling	41
2.8	Hierarchical Models	43
2.9	Summary and Outlook	44

## 2.1 INTRODUCTION

In this chapter, we attempt to give a brief overview of the following topics: (1) role of models in science, (2) statistical models, (3) Bayesian and frequentist analysis of statistical models, (4) Bayesian computation, (5) WinBUGS, (6) advantages and disadvantages of Bayesian analysis by posterior sampling, and (7) hierarchical models. This list of topics is vast and it would be impossible to give them extensive coverage even in a whole book. For topics 3–7, unless you understand the theory of

frequentist and Bayesian inference fairly well, we would expect you to also read some books or parts of books that delve more deeply in that, for instance, Gelman et al. (2004), McCarthy (2007), chapter 2 of Royle and Dorazio (2008), Carlin and Louis (2009), Ntzoufras (2009), or the introductory chapters in Link and Barker (2010). There are also useful introductory articles, such as Ellison (2004) or Clark (2005).

## 2.2 ROLE OF MODELS IN SCIENCE

Science is about rationally explaining nature by obtaining mechanistic or other explanations for the workings of a system and/or being able to predict the results of the system. However, most observable phenomena in nature are too complex for us to understand directly by simply staring at them, or rather, the system that has generated them is too complex, that is, affected by too many factors, too variable over space or time, and so on. So, explaining always requires simplifying things. Broadly, a model is nothing but a formal simplification of a complex system that we would like to explain or whose behavior we would like to predict. Indeed, at the start of this book, we have claimed that *every* interpretation of *any* observation *always* requires a model, that is, a simplification of the system, so that everybody who offers an explanation of anything has in fact a model, whether he or she knows it or not. It could also be said that explanation is impossible without a model.

So, an explanation or a more formal model is an abstraction of nature, that is, a rendition of nature with much reduced complexity. The crucial point is that we should use a good model, that is, one in which we retain the important features of the system in nature that we want to explain and only ignore the less important features. Then, by looking at this greatly simplified toy version of nature, we hopefully get a better understanding of nature herself and also can use our toy to make predictions of future or unobserved things in nature.

There are many famous sayings about models, some of which follow here. We think that they express nicely some key features of models, statistical or not:

*Modeling is as much art as it is science* (McCullagh and Nelder, 1989): this statement expresses the fact that there are not, nor can ever be, automatic, brain-free rules for building a model (although in some disciplines much effort is spent in this pursuit).

*All models are wrong, but some are useful* (Box): this is perhaps the most famous saying about models. It emphasizes that one must not look for an exact rendering of nature in a model; it is in this sense that every model is wrong. However, by simplifying, we should get some use out of a model.

Another meaning, which is perhaps not so widely appreciated, is that not all models are useful, so we should try and find the useful ones. Of course, it also begs the question of how wrong a model can be to still be useful.

*There has never been a straight line nor a Normal distribution in history, and yet, using assumptions of linearity and normality allows, to a good approximation, to understand and predict a huge number of observations* (Youden): This statement again expresses the fact that models are mere approximations, but that they can be hugely successful.

*Everything should be a simple as possible, but not simpler* (Einstein): this statement is related to the principle of parsimony and is an important guide for creating models. Very similar is the “Occam’s razor” attributed to the English logician William of Ockham, which states that the explanation of any phenomenon should make as few assumptions as possible, eliminating (or *shaving off*) those that make no difference in the observable predictions.

*Nothing is gained if you replace a world that you don’t understand with a model that you don’t understand* (we heard Maynard Smith quote this one, but he may have had it from somebody else): we like this statement because it reminds us of the importance of the principle of parsimony in modeling—“*Keep it as simple as possible*”. It also expresses the notion that we must replace a world that we *do not* understand by a model that we *do* understand, that is, typically something simpler. Of course, we could also understand this statement as a call for becoming a better modeler.

Finally, here is a claim we have made elsewhere (Kéry, 2010): *It is difficult to imagine another method that so effectively fosters clear thinking about a system than the use of a model written in the language of algebra*. There are various ways to express a model; words (language), graphs, and equations are some of them. Unfortunately, the human language is often very inadequate to express the subtle details of the multitude of potential explanations (= “models”) for a given system. Trying to put down on paper all the elements of an explanation in the language of algebra has the big advantage that it forces us to think much more clearly about the system we want to understand. One of the things we like most about the WinBUGS software ([Section 2.5](#)) is the BUGS language (Gilks et al., 1994). Describing a model in BUGS comes very close to describing it in simple algebra. So to us, describing a model in the BUGS language is one of the most transparent ways of building a model.

Conceptually, and written in algebra, a model looks something like the following:

$$y = f(x, \theta)$$

Here,  $y$  is a response, something that our study system has produced and whose genesis we would like to understand. Often, we are interested in predicting future responses produced by the study system or responses for particular values of one or several explanatory variables  $x$ . The response is a function  $f$  of one or several explanatory variables  $x$  and of some system descriptors, or parameters,  $\theta$ . Here,  $f$  would include the particular parametric form of the relationship between  $y$  and  $x$ . In essence, then, modeling means to replace a complicated reality of very large dimension with a much smaller set of system descriptors called parameters ( $\theta$ ). An explicit simplified system description in algebra is called a mathematical model.

There are broadly two different objectives of modeling, and they may lead to two different modes of building a model: explanation and prediction. Explanation means understanding and will typically require simpler models than prediction (Caswell, 1988). The explanatory mode of modeling focuses on the actual model structure. It is hoped that the kind of parameters and their values have some relevance for how nature generated the observed output. The focus is more on the parameters  $\theta$ . In contrast, prediction focuses on the system output, the response  $y$ , and thus aims at predicting the response as well as possible either within the sample studied or for the entire statistical population that is represented by the sample. Common to both modes of modeling is that we must first build a model and estimate its parameters  $\theta$ .

There is an important distinction between what might be called implicit and explicit models. We have claimed before that any interpretation of nature requires a model, but we believe that many people are not aware of this. When we talk to somebody in the general public or to not-so-modeler-types of ecologists, we often sense a certain distrust in formal, explicit modeling explanations of nature. Also, folks often have a strong feeling that the observed data are somehow superior to an inference made from these same data under a formal model. We often hear exclamations like the following: “oh yes, but this is only a model and we all know models are wrong; better stick to the data—there at least we *know* where we’re at”. On the whole, we think that the reasoning behind this feeling is flawed in the sense that *any attempt at explanation requires a model*.

Of course, it is possible to build models that have little to do with reality and are useless to understand or to predict a particular system. However, there cannot ever be a conclusion, deduction, or inference from any observation alone. Data need models, simply some people have explicit models and others have only implicit models. Implicit modelers frequently do not know that they are modelers, too, and that their conclusions are always contingent upon a certain set of assumptions.

These assumptions are usually unstated and may or may not be appropriate for any particular case. But just because you do not describe assumptions explicitly does not mean that these assumptions do not exist. Worse yet, if assumptions are not made explicit, they cannot be scrutinized. Just go and ask an implicit modeler about the goodness-of-fit of his or her explanation, that is, implicit model! So, again, everybody is a modeler, but some recognize this and some do not.

## 2.3 STATISTICAL MODELS

---

Almost anything in nature is affected by such a large number of factors that we could never measure or even identify them all. The result is that virtually any system that we encounter in nature will be stochastic, that is, its outcome is to some degree unpredictable. This means that a response is best thought of as the realization of a random variable. In colloquial language, we might say that chance is involved in the generation of our observations. Chance does not mean that something has no reason for happening, in the sense that there is no cause for it: there is always a cause, simply we do not know it and therefore cannot understand it completely or predict an observation perfectly.

In our models for explaining or predicting nature, we then need a description of the combined effects of all unknown and un- or mismeasured factors. A convenient mathematical description is by use of the concept of a random variable with a probability distribution function (pdf). For a random variable, this function assigns a probability of occurring to each element of a set of outcomes that are possible. To account for the unpredictable element in our observations, a model must incorporate a stochastic component and then a mathematical model becomes a statistical model. Our sketch of a model might then become:

$$y = f(x, \theta) + \varepsilon \quad \text{with} \quad \varepsilon \sim g(\phi)$$

Here,  $\varepsilon$  is the part of the response that is not explained by the functional form of the model  $f$ , the explanatory variable(s)  $x$ , and the parameter  $\theta$ , and  $g$  is a function describing that unexplained part using parameter  $\phi$ . A statistical model is often paraphrased as

$$\text{response} = \text{systematic} + \text{stochastic}$$

That is, we imagine that our response consists of a systematic and a stochastic part. Other pairs of terms for the same idea are deterministic + random and signal + noise. We will see later (Chapter 3) that this concept must be extended in so-called hierarchical models, where it applies separately to each component model, that is, level in a hierarchy.

## 2.4 FREQUENTIST AND BAYESIAN ANALYSIS OF STATISTICAL MODELS

One often hears the phrase “*we analyzed the data*”. We believe that data analysis should be seen as consisting of two fairly distinct activities: first, to construct a plausible model of the processes that could have produced the data we observe and second, to analyze that model, for example, to find values for its parameters or to predict what the observed data might be under specific circumstances. Of course, the two activities are intertwined, but nevertheless we think that it is useful to distinguish them conceptually. One example of where this helps is by recognizing that in a sense, there is no “Bayesian so-and-so model”. Rather, we first build a model and then, we may decide to analyze it in a Bayesian or in a classical framework. So what is the difference between a Bayesian and a classical (or frequentist) analysis?

The difference between classical and Bayesian statistics really starts with the analysis of a model, if one forgets for a moment the obvious difference that any model analyzed in a Bayesian mode of inference must contain prior distributions (see below). Frequentists and Bayesians differ in the way they treat the uncertainty about what is unknown in a model, especially the uncertainty about a parameter  $\theta$ .

For a frequentist, parameters are fixed and unknown quantities and uncertainty about them is expressed in terms of the variability of hypothetical replicate data sets produced by them. Uncertainty is evaluated over these hypothetical replicates, even if the only thing we ever have is a single data set. Probability is defined as the long-run frequency of events in such hypothetical replicates; therefore, classical statistics is often called frequentist statistics. Frequentists only make probability statements about the data, given fixed parameter values, but never about the parameters themselves, as one might want to. In other words, frequentists do not assign a probability to a parameter; rather, they ask about the probability of observing certain kinds of data given certain values of the unknown parameters. Probability statements such as standard errors refer to hypothetical replicate data that would be expected if certain parameter values hold; they are never directly about these parameters. In the frequentist world, it is impossible in principle to make a statement such as “I am 95% certain that this population is declining”.

Bayesians define probability in a fundamentally different way. Their probability is the individual belief that an event happens or that a parameter takes a specific value. No hypothetical replicates are required in Bayesian inference (though they are useful for instance in model checking; see posterior predictive checks; Gelman et al., 1996). For a Bayesian, probability is the sole measure of uncertainty about all unknown quantities: parameters, unobservables, missing or mismeasured values, or future or unobserved

responses (predictions). Bayesians use probability as their unified measure of uncertainty. This allows them to apply the mathematical laws of probability for parameter estimation and all their statistical inference. Therefore, it is possible to make probability statements about the unknown quantities, given the data, by simple use of conditional probability.

One often reads that in frequentist statistics, a parameter is a fixed and unknown quantity, while in Bayesian statistics it is a random variable. This is misleading. Rather, also for Bayesians, parameters may represent fixed and unknown quantities, but because Bayesians describe their uncertainty, or their imperfect knowledge, about the unknown parameter in terms of probability, they are *treating* parameters as random variables (Link and Barker, 2010).

So how do frequentists and Bayesians go about parameter estimation and inference? Both usually start with the sampling distribution of the data, also called the data distribution. This is the statistical description of the mechanism that could have produced the observed data, that is, the statistical model. The sampling distribution of the data  $y$  is a function of a possibly vector-valued parameter  $\theta$ . It is denoted  $p(y | \theta)$  and read “the probability of  $y$ , conditional on (given)  $\theta$ ”. An example might be that conditional on  $\theta$ , a set of counts  $y$  has a Poisson distribution:  $p(y | \theta) \sim \text{Pois}(\theta)$ . This is often abbreviated to  $y | \theta \sim \text{Pois}(\theta)$  or even  $y \sim \text{Pois}(\theta)$ .

In frequentist statistics, the likelihood function plays a central role for inference about parameters. The likelihood function is the same as the sampling distribution, but “read in reverse”: we interpret the sampling distribution of the observed data as a function of the unknown parameters  $\theta$ , with the data  $y$  fixed. This is denoted  $L(\theta | y)$  and read as “the likelihood of parameter  $\theta$ , given the data  $y$ ”. We choose as our best guess of  $\theta$  that parameter value which leads to the maximum function value when plugged into the sampling distribution function for the observed data  $y$ . The likelihood is not a probability because it does not integrate to 1, and the maximum function value may be greater than 1. Frequentists estimate a single point of the likelihood function and call the value which maximizes that function the *maximum likelihood estimate* or *MLE*. In other words, the MLE represents parameter value(s) which maximize the probability of getting the data actually observed. Any other value gives a lower probability of getting one’s data.

Here is an example for a simple maximum likelihood analysis. Let us assume we wanted to empirically determine the detection probability of tadpoles by releasing some in a small artificial pond and counting them later. Say, we released  $n = 50$  tadpoles and then count  $y = 20$  of them and we want to estimate the probability that a tadpole is seen ( $\theta$ ). The typical sampling distribution assumed for this scenario is a binomial, that is,

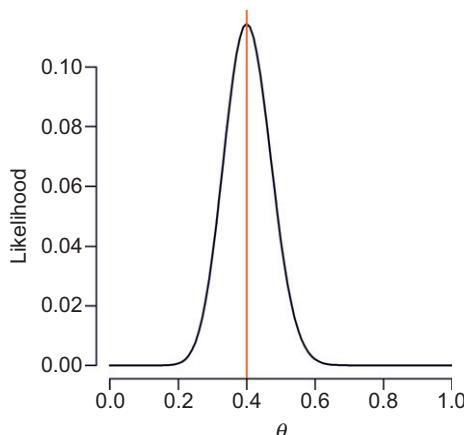
$$p(y | \theta) = \frac{n!}{y!(n-y)!} \theta^y (1-\theta)^{n-y}.$$

The method of maximum likelihood takes as the best estimate that value of  $\theta$ , which, when plugged into this sampling function along with the data ( $y$ ), yields the highest function value, that is, the highest likelihood  $L(\theta | y)$ , where  $L(\theta | y) \propto p(y | \theta)$ . Thus, we can plug in different possible values of  $\theta$  (we know that the value must be in the range from 0 to 1), compute the value of the likelihood function, and take that values of  $\theta$  for which the likelihood is maximal (Fig. 2.1). We see that the likelihood function reaches a maximum for  $\theta = 0.4$ , so this is the maximum likelihood estimate (MLE) of  $\theta$ , often denoted  $\hat{\theta}$ , and written as  $\hat{\theta} = 0.4$ .

In this simple case, it is possible to obtain the MLE analytically. This requires that we calculate the first derivative of the likelihood function, set it to zero, and solve the equation with respect to  $\theta$ . Since the binomial coefficient (the ratio of factorials just after the equal sign above) is a constant, we do not need to include it in this calculation. Thus, we have

$$\begin{aligned} L(\theta | y) &\propto \theta^y (1 - \theta)^{n-y} \\ L(\theta | y) \partial \theta &= \theta^y (1 - \theta)^{n-y} \left( \frac{y}{\theta} - \frac{n-y}{1-\theta} \right) \\ \theta^y (1 - \theta)^{n-y} \left( \frac{y}{\theta} - \frac{n-y}{1-\theta} \right) &= 0 \\ \hat{\theta} &= \frac{y}{n} \end{aligned}$$

We see that the ratio 20/50 is the MLE of  $\theta$ , which is what we have expected.



**FIGURE 2.1** Binomial likelihood function for detection probability ( $\theta$ ) in the tadpole example, where 20 of 50 released tadpoles were seen. The MLE of  $\theta$  is 0.4.

MLEs have several desirable features, such as asymptotic unbiasedness, consistency, and invariance to transformation (see, e.g., chapter 2 in Royle and Dorazio, 2008, or any book about mathematical statistics). However, the method of maximum likelihood is based on asymptotic approximations; for instance, MLEs are only unbiased and associated standard error estimates valid when sample size goes to infinity. Every ecologist knows that we rather rarely have infinite samples in ecology. How well MLEs and their standard errors perform in the typical small sample size situations in ecology is an open question in any actual application (Le Cam, 1990).

In contrast, the basis for Bayesian inference is the so-called Bayes rule. Bayes rule is attributed to the seventeenth century English minister and mathematician Thomas Bayes (Bayes, 1763). It is an undisputed mathematical fact which is easily proven from the rules of probability. Consequently, not every application of Bayes rule makes an analysis Bayesian: the application of Bayes rule to observables is undisputed. However, what Bayesians do is to apply Bayes rule also to unobservable quantities, such as, most importantly, the parameters in a statistical model. As Lindley (1983, p. 2) put it so succinctly, the recipe for every Bayesian analysis about any uncertain quantity is quite simple and mechanical:

- *What is uncertain and of interest to you? Call it  $\theta$ .*
- *What do you do know? Call it  $D$  [...].*
- *Then calculate  $p(\theta | D)$ .*
- *How? Using the rules of probability, nothing more, nothing less.*

To describe how Bayesians learn from data using the rules of probability, we will introduce Bayes rule in the context of two sets of mutually excluding, *observable* events,  $A$  and  $B$ . Remember that a vertical bar ( $|$ ) means “conditional on” and is read as “given”.

$$p(A | B) = \frac{p(B | A)p(A)}{p(B)}$$

This says that the conditional probability of observing  $A$ , given that  $B$  has happened or is true,  $p(A | B)$ , is equal to the conditional probability of observing  $B$  given  $A$ ,  $p(B | A)$ , times the marginal probability of  $A$ ,  $p(A)$ , divided by the marginal probability of  $B$ ,  $p(B)$ . To better see how Bayes rule works for statistical learning from data, consider the following example which is inspired by a similar example in Pigliucci (2002). Assume that our activity after work consists of bird watching ( $B$ ) or watching football on TV ( $F$ ) and that this depends on whether the weather is good ( $g$ ) or bad ( $b$ ) on a particular night. Let us assume that you knew the following: the joint probability of good weather and us watching birds is 0.5, the marginal probability of good weather is 0.6 and the marginal probability of us watching birds is 0.7. Now if you are told that we were watching football on a particular night, what is your best guess about the weather that night?

For illustration, we present all involved probabilities in a two-by-two table with margins added (Table 2.1). In this example, we deal with mutually exclusive events (e.g., the weather cannot simultaneously be good and bad); hence, probabilities must add up within the same rows and columns, respectively. We also know that the four main cells must sum to 1, so we can fill in all cells in the table from the information just given.

Note that  $p(A, B) = p(A | B)p(B) = p(B | A)p(A)$ : the joint probability that  $A$  and  $B$  both occur is equal to the product of the probabilities that  $A$  occurs given that  $B$  has occurred and that  $B$  occurs, and vice versa. We had asked for your best guess about the weather on a night we were watching football. As a response, we can compute  $p(b | F)$ , the conditional probability of bad weather, given that we were watching football:

$$p(b | F) = \frac{p(b, F)}{p(F)} = \frac{0.2}{0.3} \approx 0.66$$

So at the outset, without any additional information, your best guess of the probability of bad weather that night would simply have been the marginal probability  $p(b) = 0.4$ . However, given that we watch football more frequently when the weather is bad, the knowledge that on that particular night we were watching football has increased your best guess at the probability of bad weather from 0.4 to 0.66.

This example illustrates how the information about our postwork activity ( $D$  in Lindley's recipe) influences our knowledge about the weather on a given night. In other words, it shows how our prior knowledge about the weather,  $p(\theta)$ , was updated by the observed data  $D$  to become  $p(\theta | D)$ . This example deals with observable events of a binary nature and nicely illustrates the use of conditional probability for learning from data, which is the basis for using Bayes rule for statistical inference about unknown quantities. In Bayesian inference, parameters take the place of the weather in our example and the data correspond to the knowledge about our

**TABLE 2.1** Joint and Marginal Probabilities of Events for Two Sets of Mutually Exclusive Events, Bird Watching/Watching Football and Good/Bad Weather

	Good Weather (g)	Bad Weather (b)	
Go Bird Watching ( $B$ )	<b>0.5</b>	0.2	<b>0.7</b>
Watch Football ( $F$ )	0.1	0.2	0.3
	<b>0.6</b>	0.4	1.0

*Note:* The probabilities given in the text are printed in bold face and the remainder can be obtained by simple addition and subtraction.

postwork activity. In addition, we use Bayes rule to assess the uncertainty about both discrete events and continuous quantities.

When Bayes rule is applied to statistical inference about parameter  $\theta$  based on the information in the data  $D$ , the probability  $p(\theta|D)$  is called the posterior distribution of  $\theta$ : it is the conditional probability of parameter  $\theta$ , given the (known) data  $D$ , the prior and the model:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$$

There are three further quantities in Bayes rule, apart from the posterior distribution  $p(\theta|D)$ . In some ways,  $p(D|\theta)$  is the opposite of the posterior: it is the probability of the data, given the parameters, or, as used here, the likelihood function. It may appear confusing that the likelihood in Bayes rule is traditionally written as  $p(D|\theta)$ , that is, in the same way as the sampling distribution of the data. Quantity  $p(\theta)$  is the probability of the parameters, that is, the prior distribution. Finally,  $p(D)$  is the marginal probability of the data and is defined as the integral of the numerator over  $\theta$ . This is a constant used to normalize the right-hand side of Bayes rule so that the result integrates to one and becomes interpretable as a probability. As an aside, we note that it would be wrong to say that Bayesian inference is not likelihood based. Obviously, the likelihood function is a central part of Bayesian inference.

Following up the tadpole example from above, how would we estimate the unknown parameter  $\theta$  in the Bayesian framework? Well, we have already defined the likelihood function. We now need to define a prior distribution of  $\theta$ , that is, specify what we know *a priori* about  $\theta$  and express this knowledge in a probability distribution. We know that  $\theta$  must lie between 0 and 1. A useful probability distribution defined on the interval  $(0, 1)$  is the beta distribution with parameters  $\alpha$  and  $\beta$ . By specifying values of  $\alpha$  and  $\beta$ , we can express our knowledge about  $\theta$ . Generally, we have

$$p(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}.$$

Note that we have excluded a constant of the beta distribution as it has no relevance for estimating  $\theta$ . Having chosen likelihood and prior, we can analytically obtain the posterior distribution:

$$\begin{aligned} p(\theta|y) &\propto p(y|\theta)p(\theta) \\ p(\theta|y) &\propto \theta^y(1-\theta)^{n-y}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ p(\theta|y) &\propto \theta^{y+\alpha-1}(1-\theta)^{n-y+\beta-1} \end{aligned}$$

We see that this posterior distribution is also a beta distribution, with mean  $\frac{y+\alpha}{n+\alpha+\beta}$  and mode  $\frac{y+\alpha-1}{n+\alpha+\beta-2}$ .

Absent any prior knowledge about  $\theta$ , we would specify  $\alpha = 1$  and  $\beta = 1$  because this would result in a uniform prior distribution for  $\theta$ , representing a belief that any value of  $\theta$  between 0 and 1 is equally likely. A prior which says that we do not know anything about the parameter (or do not care about what might be likely values) is called noninformative, vague, flat, or diffuse prior. In the tadpole example, the resulting posterior mean is 0.404 and the mode is 0.400. The posterior mean is very close to the MLE and the mode is exactly the MLE, as we would expect—with a noninformative prior, the posterior mode of a parameter in a Bayesian analysis corresponds to the MLE of that parameter in a frequentist analysis. We notice that the impact of the prior (the values of  $\alpha$  and  $\beta$ ) on the posterior distribution diminishes with larger sample size ( $n$ ). The posterior distribution can also be plotted and functionals (e.g., probability that  $\theta > 0.5$ ) may be computed (see later).

So Bayesian statistical analysis is conceptually very simple: probability (via Bayes rule) is the basis for all inference about parameters and any other unknown quantities in a system analyzed. Bayesian statistics has great philosophical appeal (Link and Barker, 2010) since it is conceptually so simple (all inference is based on Bayes rule), exact (e.g., standard errors are those for your actual data set and not for some infinite version of it), and coherent (logically consistent).

Bayes rule is often paraphrased like the following:

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

that is, the posterior distribution is proportional to the product of the likelihood and the prior. This makes it clear that in a Bayesian analysis, one's conclusion, that is, the posterior distribution, is very openly *always* a result of both the information contained in the data (as embodied in the likelihood function) *and* of our prior knowledge (our assumptions) about the unknowns in the model. We cannot conduct a Bayesian analysis without formally expressing our *a priori* uncertainty/knowledge about the parameters in the form of a probability distribution.

Several other points are noteworthy about Bayes rule as a basis for inference. First, Bayes rule formalizes the way in which humans learn. Learning always consists of updating what we knew before with what we see now. Bayes rule is thus a mathematical formalization of how we deal with new information: we always weigh the information of any new observation with the knowledge, or prior experience base, that we possessed before making that observation. The result, our conclusion, is then affected by both, and the relative importance of one or the other can vary. For instance, if we know something for almost certain, we would require large quantities of data to overthrow that prior belief. In contrast, if we do not know anything at all about a system, we might be happy to draw a conclusion based on very little data. This conclusion would then be the result almost entirely of the new

data. In Bayes rule, this weighting of information happens in a formal and mathematically rigorous way.

As another illustration of this point, note that in virtually every analysis in ecology we know something about the system analyzed and we always use that information, even in a frequentist framework. For example, if we get parameter estimates that seem to make no sense when compared with what we think we know about the system (“results do not make sense biologically”), many of us are prepared to dismiss these results in an act of *ad hoc* Bayesianism. In contrast, in a proper Bayesian analysis, prior expectations about the results could be formally introduced into an analysis. Such expectations really amount to available knowledge that is not formally used otherwise in a frequentist analysis, and this does not seem a very sensible thing to do, if we think about it.

Second, Bayes rule shows us how we can combine several pieces of information in a mathematically rigorous manner. Simply treat one piece of information as prior information and the other piece of information as the data, form a likelihood for the latter, apply Bayes rule and out comes your combination of the information in the form of the posterior distribution.

Third, priors can simply be seen as assumptions. Hence, Bayes rule represents an instrument by which we can compare formally the effect of different assumptions about model parameters by repeating the calculations with different priors.

Finally, and fourth, Bayes statistics is normative in the sense that it prescribes a mathematically rigorous way of arriving at a logical conclusion from data, a model, and *a priori* assumptions (Lindley, 2006). A Bayesian will not argue about what prior assumption one should make; this is really in the realm of the subject-matter scientist. However, once people have decided on their priors, then Bayes rule *prescribes* a mathematically rigorous way in which our statistical conclusions ought to be drawn from some observed data, using a model and these prior assumptions.

Hence, one might think that the ability to specify prior distributions was widely regarded as an advantage of the Bayesian approach. However, interestingly, priors are more often viewed as a liability of Bayesian analyses, for several reasons (Dennis, 1996). The first reason is that priors need to be decided upon. This choice is somewhat subjective even if based on past data because whether these data are relevant for the current analysis may be debatable. Hence, Bayesian analysis is intrinsically subjective (but at least explicitly so, one might want to add). Many people feel uneasy at making an explicit decision about what might be plausible values for a parameter and find it difficult to make a choice for the prior distributions.

Second, if two persons use different priors for their analysis of the same data set, they may clearly get different answers because the posterior is always the result of combining the information in the data with that in

the prior. Worse yet, even if both agreed to specify vague priors, which does not contain information, they might still end up with different answers under two different such vague priors. So, even vague priors can be challenging because different forms of specifying absence of knowledge about a parameter might not be equal. For instance, for a parameter representing a probability, we might use a uniform distribution on the interval 0–1 to say that any value is equally likely. When we specify that same parameter on the logit scale as we often do (see Chapter 3), then something analogous would be to use a uniform distribution with a large range, for example, from –1000 to 1000. Although the two prior distributions are both vague on their scales, the posterior distributions will not be exactly the same.

In spite of all this, it must be said that it is easy to exaggerate prior-related difficulties with the Bayesian approach. For once, and perhaps to console some doubters, typically parameter estimates from the Bayesian analysis of a model with vague priors numerically match pretty closely the MLEs from a frequentist analysis of the model. Second, with reasonable sample sizes, the data overwhelm the prior in their influence on the posterior distribution because the effect of the prior diminishes as sample size increases. Data cloning, a method to use Markov chain Monte Carlo simulation (see Section 2.5) to obtain MLEs without effects of priors, is based on this fact (Lele et al., 2007). Third, in any Bayesian analysis, it is customary to report the priors used. If an analyst disagrees with the choice, he or she could—at least in principle—repeat the analysis with his or her favorite prior. Finally, it is a good practice (though far from always done) to try out several priors and see what their effect on the inference is, that is, do a prior sensitivity analysis and report its results.

In this book, we follow Royle and Dorazio (2008), and indeed most applied Bayesian analysts, and specify vague priors for a natural parameterization of a model. For a parameter representing a probability, we often use a uniform(0, 1) or beta(1, 1) distribution or adopt a uniform distribution with a suitably wide range (e.g., –10, 10) for the same parameter on the logit scale. Alternatively, we often specify a flat normal distribution, that is, a normal distribution with suitably large standard deviation. What represents a suitably wide range or large standard deviation depends on the support of the likelihood function. If the likelihood of a parameter is essentially zero outside of, say, 0.4–0.6, then a uniform prior with a range between 0 and 1 may be sufficient to not affect the posterior distribution. On the other hand, if the likelihood function has nonnegligible support over a larger range, a uniform prior intended to be vague must also have a wider range. Whether a prior is sufficiently vague may be easily ascertained by repeating an analysis with narrower or wider priors and seeing whether the posterior is affected or not. In the case of a uniform prior, a posterior distribution that is truncated by either or both limits

shows that the analyst has not succeeded in choosing a vague prior. Consequently, a wider range must be chosen.

For variance parameters, the typical prior chosen used to be an inverse gamma distribution for a long time. However, currently, the preferred choice of many Bayesian analysts seems to be a suitably wide uniform for the variance parameter on the scale of the standard deviation (Gelman, 2006). This is our typical choice. We do not usually adopt explicitly informative priors. However, when the Markov chains (see [Section 2.5](#)) for a parameter fail to converge, we may narrow the range of a uniform prior or reduce the standard deviation of a flat normal prior to achieve convergence.

We saw that the basis of all Bayesian inference was the posterior probability distribution of the parameters. So once we have that, what should we do with it?

In special cases, we might choose to plot the posterior distribution for an important parameter. However, in most cases, it will be enough to summarize the posterior distribution for some or all model parameters by reporting its central tendency and its spread. Typically, for a point estimate, the posterior mean, median, or mode is used. With vague priors, the posterior distribution reflects the likelihood function directly; hence, the posterior mode is equivalent to the MLE of a corresponding frequentist analysis. The standard deviation of the posterior distribution is analogous to the standard error of a parameter estimate in a frequentist analysis. Any interval which contains 95% of the posterior mass is a Bayesian analogue to the frequentist confidence interval (CI) and is usually called a credible interval (CRI), or sometimes also a Bayesian confidence interval. Often the 2.5th and 97.5th percentiles of the posterior samples are taken as a 95% CRI; this is what we do in this book.

At this stage, and especially because posterior-based Bayesian parameter estimates often very closely match their MLE analogs numerically, it is important not to forget the exact meaning of these quantities. This has to do with the different definitions of probability. For instance, a 95% frequentist CI does *not* contain the target parameter with probability 0.95. In frequentist statistics, probability statements are about the data, or in this case, about the method, and never about the parameters. Hence, the 95% refer to the reliability of the method of constructing a 95% CI. If we sampled data from the same population 100 times and for each formed a 95% CI for a certain parameter, then about 95 intervals would indeed contain the population value and another 5 would not.

In contrast, a Bayesian 95% CRI *does* contain the parameter with probability 0.95. Also, we can make other probability statements about parameters, for instance, of the kind “I am 92% sure that this population is declining”, by looking at the proportion of the mass  $r < 1$  of the posterior distribution for a population growth rate  $r$ . Or else, “I am 50% sure that the growth rate lies between 0.5 and 0.8”. This is a great asset of a Bayesian analysis,

especially when describing the results to the public or resource managers. The Bayesian definition of probability (and especially of uncertainty intervals) conforms much more closely to the human concept of probability than the repeated-sample definition in frequentist statistics.

## 2.5 BAYESIAN COMPUTATION

---

To analytically evaluate the posterior distribution, solving Bayes rule for all but the simplest models involves high-dimensional integrations, which can be very difficult or actually impossible to solve in most cases. The tadpole example above (Fig. 2.1) is a fairly simple example which is not difficult analytically. Most posterior distributions are, however, much more complicated, and no closed-form formulas exist. Hence, up to about 20 years ago, Bayesian analysis of a more complex model was typically not really an option. However, at the beginning of the 1990s, some statisticians rediscovered pioneering work done by physicists back in the 1950s (Smith and Gelfand, 1993). This work showed that simulation techniques could be used to draw samples from the posterior distribution instead of solving the equations. Specifically, Metropolis et al. (1953) and Hastings (1970) developed so-called Markov chain Monte Carlo (MCMC) algorithms. MCMC yields samples of arbitrary size of dependent (i.e., autocorrelated) draws from a distribution and can be constructed so that this distribution approximates the desired posterior distribution. Hence, these samples can be summarized for inference about the posterior distribution; for instance, mean and standard deviation of the samples can be interpreted as the posterior mean and posterior standard deviation, that is, as a Bayesian point and interval estimate of a parameter. Samples can also be plotted, in a raw or a smoothed histogram, for a picture of the posterior distribution.

The rediscovery and successive refinement of MCMC algorithms, along with the ever-increasing power of personal computers, sparked a revolution in statistics and also in the empirical sciences such as ecology (McCarthy, 2007). This revolution is still going on and has catapulted Bayesian methods to the center of the ecological data analysis scene (Brooks, 2003). However, for most ecologists, constructing their own MCMC algorithms would be prohibitively difficult. Hence, the Bayesian revolution has only recently reached ecology.

## 2.6 WinBUGS

---

What has brought the Bayesian revolution to ecology has a name: WinBUGS (Gilks et al., 1994; Lunn et al., 2000, 2009). WinBUGS is the Windows version of a free computer program developed as part of the

BUGS project, which means *Bayesian inference using Gibbs sampling*; see [www.mrc-bsu.cam.ac.uk/bugs](http://www.mrc-bsu.cam.ac.uk/bugs). WinBUGS originally used a particular variant of MCMC called Gibbs sampling (Geman and Geman, 1984), but now uses a variety of other MCMC sampling techniques. For a history of the BUGS project, an appreciation and outlook, see Lunn et al. (2009). The active development in the BUGS project now takes place with OpenBUGS; see [www.openbugs.info](http://www.openbugs.info). At the time of writing, the two BUGS sisters are nearly identical in practice and will run most code from one another fine. The same goes for a BUGS clone called JAGS (*Just another Gibbs sampler*, see [www-fis.iarc.fr/~martyn/software/jags](http://www-fis.iarc.fr/~martyn/software/jags)). JAGS is another MCMC engine that uses the BUGS language; hence, most BUGS code should run also in JAGS.

For most ecologists, WinBUGS is simply an ingenious MCMC blackbox. The analyst communicates with the MCMC engine by providing a data set and describing a statistical model for it using a simple and effective model definition language, the BUGS language (Gilks et al., 1994). In our opinion, the BUGS language can claim a large part of the value for ecologists of the WinBUGS software. All statistical models that we have had to do with are specified more simply and—to us—in a *much* more transparent way in the BUGS language than when using custom code for maximum likelihood estimation. For the latter, one needs to define the likelihood for a model explicitly and then use some function optimizer (for instance, `n1m` or `optim` in R) to find the MLEs (Bolker, 2008). Alternatively, one uses software that shields us from most of the complexity but makes it easy to fit models that one does not understand or that may not make sense. In contrast, BUGS code often looks trivially simple and concise. In the BUGS language, all stochastic models are described by specifying local stochastic or deterministic relationships between quantities such as parameters and data. By breaking apart an entire model into its smaller component parts, understanding is greatly enhanced for an ecologist. Moreover, the construction of even very complex models becomes relatively feasible and transparent. BUGS model descriptions are naturally hierarchical, and indeed, WinBUGS is ideal for fitting hierarchical models (see Section 2.9). We will see many examples for this later in the book. Indeed, we have found that WinBUGS frees the creative modeler in many ecologists.

Once the model is specified, WinBUGS constructs an MCMC algorithm in perfect blackbox manner and runs that for the required length. Its primary product is a long stream of numbers, one for each model parameter that we choose to estimate. If the MCMC algorithm has been constructed adequately and the chains have converged to the desired posterior distribution, then these numbers represent a random sample from these posterior distributions. There is an autocorrelation built into these numbers, since they form a Markov chain. This means

that the first part of the chains, where the effect of the arbitrarily chosen starting values will still be felt, must be discarded as a so-called burnin. Whether the burnin period is over or not can be judged by visual means, that is, by inspecting a time-series plot of the sampled values for each parameter. The plot should now randomly jump up and down around a constant mean. There are formal criteria to decide whether convergence has been reached. For instance, the Brooks–Gelman–Rubin statistic (Brooks and Gelman, 1998) is often used. It requires two or more chains for each parameter and compares the between-chain with the within-chain variance in an ANOVA fashion. At convergence, the value of this test statistic, sometimes called Rhat, is 1. After a chain has converged onto the desired target distribution, to save computer space and reduce autocorrelation, one may thin it by  $k$ , that is, keep every  $k$ th value only. Thus, one gets a smaller, but more information-dense (because less autocorrelated) sample from the posterior distribution.

WinBUGS can be used as standalone software, see McCarthy (2007), Ntzoufras (2009) or chapter 4 in Kéry (2010). However, we find it more efficient to harness it to R via the communicator package R2WinBUGS (Sturtz et al., 2005). This is how we use WinBUGS throughout this book.

There are many Bayesian statistics books that explain MCMC algorithms and give examples (e.g., McCarthy, 2007; Ntzoufras, 2009; Link and Barker, 2010; King et al., 2010); therefore, we skip this here. We feel that the importance of being able to code one's own MCMC algorithm may easily be overstated. After all, hardly any ecologist would nowadays be able to code a Newton–Raphson algorithm for fitting a GLM or a Laplace approximation for the integrals that need to be solved for obtaining mixed-model estimates. And yet, many of us routinely use these methods for our research.

Admittedly, MCMC may be somewhat more difficult than these techniques and may fail in perhaps more ways than other computing algorithms commonly used for a frequentist analysis. We do not doubt that it can be a great advantage to actually *know* how to code MCMC algorithms, not least because custom-written MCMC code often runs much faster than WinBUGS. Nevertheless, a simple intuitive understanding of the nature of MCMC techniques will often be enough for ecologists. Such an understanding may be obtained by simply using an MCMC blackbox, such as WinBUGS and experiencing the behavior of the chains in many situations for many different models. This is what we do in this book.

WinBUGS is a fantastic program, but may exhibit a fair dose of pretty idiosyncratic behavior. There are many things that one just must know in order to succeed. A collection of survival tips can be found in Appendix 1.

## 2.7 ADVANTAGES AND DISADVANTAGES OF BAYESIAN ANALYSES BY POSTERIOR SAMPLING

There are many advantages of the Bayesian analysis of a model by posterior simulation via MCMC techniques (Kéry, 2010). Some of them which are particularly relevant for an ecologist include the following:

- Even difficult models can be fit, including some which cannot be fitted in the classical framework.
- Derived quantities may be computed trivially easily, with full propagation of the uncertainty in the components that make up the derived quantity. This can be a very hard problem in the classical framework.
- All results are exact; there are no asymptotics involved in the estimates as for MLEs, which may be of questionable value in small-data situations so typical of ecological studies.
- The BUGS language allows the typical quantitative ecologist to actually understand the construction of even complex models so that the code can be modified to fit one's own purposes.

On the other hand, there are also challenges with the Bayesian approach. At first sight, like any new theory or method, Bayesian statistics may appear difficult. Then, the choice of priors and the sensitivity of the estimates to that choice need some thinking. MCMC engines such as WinBUGS are blackboxes and are hard to understand, leaving a certain uneasiness with people who like to understand most of what they do, and convergence of the Markov chains may be difficult to assess.

MCMC-based analyses in general can be slow compared to other ways of model fitting (see, e.g., the comparisons in Kéry, 2010). Just because WinBUGS is an extremely flexible, generic MCMC engine, it is a rather slow software when compared with custom-written MCMC algorithms. For complex models applied to large data sets, WinBUGS may become too slow to be of practical value. Novel algorithmic techniques that provide approximate analytical solutions to the integrations involved in Bayesian analysis (e.g., AD model builder; see <http://admb-foundation.org/> or R-INLA, Rue et al., 2009) may then be exciting new avenues for the fitting of some classes of hierarchical models.

Some other difficult topics include the detection of parameter identifiability and model selection. Lunn et al. (2009) say that the flexibility of WinBUGS to specify even very complex models may let the user fit models that do not make sense, for instance, models with parameters that are not identifiable, that is, for which the data do not contain any information. Nonidentifiability of a parameter is often difficult to diagnose in complex models, but perhaps harder still in a Bayesian than in a frequentist analysis.

This is due in part because in a sense, the problem does not really exist in the Bayesian mode of analysis: if there is no information about a parameter in the data (the likelihood), then there is always information coming from the prior, and we still technically get a posterior distribution for that parameter. Hence, one way of checking whether a parameter is indeed identified is by comparing the prior with the posterior and seeing whether changing the prior induces large changes in the posterior (Gimenez et al., 2009b; see Section 7.9). Simulating a data set and seeing whether the analysis is able to recover estimates that resemble the known input values is perhaps one of the best ways for an ecologist to check for nonidentifiability of a parameter (e.g., Schaub, 2009).

Another big topic is model and variable selection. In the frequentist world, many ecologists use model selection criteria such as Akaike's information criterion (AIC; see review by Burnham and Anderson, 2002). Yet, it appears sometimes as if a bunch of models is thrown up into the air in the hope that AIC will do all the work of sorting through them. That such a view is overly simplistic becomes clear when reading through a review paper on model selection in the primary statistical literature (e.g., Kadane and Lazar, 2004). Model and variable selection are deep waters and even among statisticians there is no consensus view on what is the best—and practically feasible—approach. Furthermore, model selection using the AIC is an unsolved problem for mixed models due to the challenge of counting the effective number of parameters (Link and Barker, 2010). Hence, for mixed models even in the classical arena, there does not seem to be a simple approach available.

These challenges appear, if anything, even more acute when one moves to a Bayesian analysis. There is an AIC-analogue called deviance information criterion or DIC (Spiegelhalter et al., 2002), but again, its standard version computed by WinBUGS appears to be problematic for hierarchical models—and most models that ecologists nowadays want to fit have more than one random component and therefore are mixed, or hierarchical, models (see Chapter 4). The DIC can be computed for such hierarchical models (see Millar, 2009, which includes R code), but the required computations are involved and computationally very demanding. Hence, in spite of long-standing criticisms of stepwise model selection and model selection by significance tests, one may effectively be back at one of those. We can look at the significance of a parameter by checking whether its 95% CRI covers zero and based on that decide whether it is warranted in a model or not. This is what we sometimes do. Other times, we simply fit one model that is biologically plausible to us and stick to that. There are yet other approaches, for instance, Bayes factors and reversible jump Markov chain Monte Carlo (RJMCMC); see for instance, King et al. (2010) and Link and Barker (2010). But be warned, Link and Barker's chapter 7 on Bayesian multimodel selection is not easy reading. Also see

the overview by O'Hara and Sillanpää (2009). It is likely that we will see more work in the future on Bayesian model selection and hypothesis testing.

## 2.8 HIERARCHICAL MODELS

In hierarchical models, complex stochastic systems are decomposed into a dependent sequence of simpler submodels. This partitioning is beneficial for a better understanding of a system, for an honest accounting for all levels of uncertainty or for computational ease. A model can be hierarchical in two ways, statistically and conceptually.

In a purely statistical sense, a hierarchical model is composed of a sequence of random variables, with the realization of the random variable at one level being a parameter of the random variable at the next level down. For instance, a hierarchical model with two levels is (dropping indices):

1.  $x \sim f(\omega)$
2.  $y \sim g(x, \theta)$

That is, at level 1 of the hierarchy,  $x$  is a realization of a random variable described by probability distribution  $f$  with parameter  $\omega$ . At level 2,  $y$  is a realization of another random variable described by probability distribution  $g$ , which depends on the realization of the first random variable,  $x$ , and on another parameter,  $\theta$ . Of course, there may be more than two levels in a hierarchical model. Hierarchical models abound in ecology. For instance, a nested ANOVA model (Kéry, 2010) is an example of a hierarchical model with two levels, with  $f$  and  $g$  being a normal distribution, such that  $x \sim N(\mu, \sigma_x^2)$  and  $y \sim N(x, \sigma^2)$ , where  $\mu$  is the grand mean,  $\sigma_x^2$  the variance among group means  $x$ , and  $\sigma^2$  the residual variance of measurements  $y$  around the group means (note indices have been omitted for clarity).

In the context of hierarchical models for population analysis, Royle and Dorazio (2008) make the important distinction between *implicit* and *explicit hierarchical models*. Explicit hierarchical models have random variables or parameters with an explicit ecological interpretation, while implicit hierarchical models do not. As an example of an implicit hierarchical model, the Poisson GLMMs in Chapter 4 have a quantity called the expected count ( $\lambda$ ). This is not a real ecological parameter because it is the product of population size and detection probability. In contrast, in the hierarchical models in Chapter 6,  $N$  is the sum of the latent indicator variables  $z$  and corresponds exactly to the local population size. In explicit hierarchical models, the lowest level in the hierarchy typically represents an explicit description of the binomial observation process. As a result, the ecological parameters in the model become directly interpretable and do have an ecological meaning. In addition, inference from explicit hierarchical models

is protected against possible misinterpretations due to a confounding of the ecological and the observation processes in implicit hierarchical models. All else equal, we prefer explicit over implicit hierarchical models.

There is another, conceptual sense in which a model can be hierarchical, and this has to do with exactly this accounting for the observation process. For example, the CJS model fitted via the m-array (Section 7.10) is not a hierarchical model in the statistical sense of the term, but the state-space version of the model (Section 7.2) is. With the m-array, the hierarchical genesis of the observed data is lost by aggregation when creating the m-array. Nevertheless, this model is still an explicit hierarchical model in a conceptual sense because its parameters have an explicit ecological meaning owing to the explicit modeling of the observation process. Similarly, the N-mixture model (Chapter 12) is intrinsically hierarchical, even when fit in a frequentist framework, where the latent abundance states are integrated out from the likelihood and thus the hierarchy is collapsed (Royle, 2004c).

## 2.9 SUMMARY AND OUTLOOK

In this chapter, we have briefly reviewed statistical models and their analysis in WinBUGS. We have claimed that any interpretation of data requires a model, either an implicit or an explicit one. Then, we briefly reviewed two philosophies for formal learning from data, with their associated methods for fitting models to data and making inferences about their parameters, that is, of obtaining estimates of the parameters and of the uncertainty around these estimates. One is maximum likelihood and the other is Bayesian inference. Bayesian inference is based on the posterior distribution, which is a product of the likelihood (representing the information contained in the data) and the prior distribution (representing what is known about the parameters beforehand). Bayesian inference uses a fact of conditional probability, Bayes rule, to let the data update our prior state of knowledge to the posterior state of knowledge. In this way, what we learn from data, the posterior distribution, is a weighted average of the prior distribution and the information of the data at hand.

We have seen that priors can be regarded both as an asset and as a liability in Bayesian inference (of course, we believe that the former outweigh the latter). We have also seen that the results of a Bayesian analysis based on the posterior distribution are much more easily explained to the public owing to the more intuitive Bayesian definition of probability. Bayesian analysis in practice nowadays means obtaining samples from the posterior distribution by simulation techniques such as Markov chain Monte Carlo (MCMC). The free WinBUGS software (along with its “sisters”, OpenBUGS and JAGS) is the most widely used MCMC engine currently available.

It allows us to specify almost arbitrarily complex models using an ingenious and simple model definition language. It then constructs an MCMC algorithm, runs it for the requested length, and produces a stream of numbers which, if all went well, represents a sample from the posterior distribution of interest. These samples can be summarized for inference, for instance, posterior means and standard deviations are customarily treated as Bayesian point and interval estimates. Important advantages of the Bayesian model fitting by posterior sampling include the numerical tractability of even very complex models, exact rather than asymptotic inference, and the ease with which derived parameters can be estimated with full propagation of all uncertainty. Some disadvantages are that it may be difficult at first, it may be slow, and parameter nonidentifiability and model selection may be even harder challenges than in the frequentist framework.

We are now armed with the motivation for population analysis and have a basic understanding for how estimation and inference about model parameters is achieved in the Bayesian framework of statistics. Hence, we are ready to move on to see our first population models. In the next two chapters, we will deal with simple models for time series of counts. Importantly, these chapters will also provide an introduction to what may be the three most essential topics of applied statistical modeling: linear and generalized linear models in Chapter 3 and random effects in Chapter 4. You will meet these concepts over and over again in your statistical modeling. If you understand them in a simple model, your understanding for more complex models will be greatly enhanced.

## 3

# Introduction to the Generalized Linear Model: The Simplest Model for Count Data

## OUTLINE

3.1	Introduction	48
3.2	Statistical Models: Response = Signal + Noise	48
3.2.1	The Noise Component	48
3.2.2	The Signal Component	49
3.2.3	Bringing the Noise and the Signal Components Together: The Link Function	54
3.3	Poisson GLM in R and WinBUGS for Modeling Time Series of Counts	55
3.3.1	Generation and Analysis of Simulated Data	56
3.3.2	Analysis of Real Data Set	64
3.4	Poisson GLM for Modeling Fecundity	66
3.5	Binomial GLM for Modeling Bounded Counts or Proportions	67
3.5.1	Generation and Analysis of Simulated Data	68
3.5.2	Analysis of Real Data Set	70
3.6	Summary and Outlook	71
3.7	Exercises	72

## 3.1 INTRODUCTION

The generalized linear model (GLM) extends the concept of a linear model from normal response models, such as analysis of variance and regression, to many other response distributions, including the Poisson and the binomial. The GLM is a crucial piece in our lego collection of modeling parts, and perhaps *the* crucial piece. Described synthetically in 1972 by Nelder and Wedderburn (also see McCullagh and Nelder, 1989; Dobson and Barnett, 2008), the GLM unifies a huge number of superficially different analytical techniques, such as regression, analysis of variance, log-linear models, and logistic regression, among many others. We believe that a solid practical understanding of the GLM is essential for the work of every serious ecologist. To understand the GLM, you must know how to build a design matrix, what a link function is and how to choose a statistical distribution for an observed response. WinBUGS, with its lucid and elegant BUGS language, is perhaps the best software available to make one really understand the GLM. Useful introductions to the GLM with WinBUGS include Gelman and Hill (2007), Ntzoufras (2009), and Kéry (2010). In this chapter, we introduce two common GLMs for count data: Poisson and binomial GLMs. The essential difference between the two is that in the Poisson GLM, counts are unbounded in principle, whereas in a binomial GLM, counts are bounded by the so-called binomial totals. The GLM is the quintessential statistical model, so we first review the concept of a statistical model as a response being composed of signal plus noise.

## 3.2 STATISTICAL MODELS: RESPONSE = SIGNAL + NOISE

The basic form of a statistical model is such that we imagine a response as being composed of two components: signal and noise. The main difference between a statistical model and a purely mathematical one is that the statistical model contains a description of the random variability in an observed response, the noise. Alternative names of the signal–noise distinction are random and fixed part of a model, stochastic and systematic part, and mean and variance, or dispersion, structure of a model.

### 3.2.1 The Noise Component

The defining feature of a statistical model is that it accommodates the randomness and unpredictability that is the hallmark of all empirical data, even those in physics, chemistry, or molecular biology labs. To describe this noise, we use statistical distributions. Our quantitative descriptors of the random part of a model, the noise component, are the parameters

of these distributions and of course their identity. Some of the most frequently used distributions in population ecology are Poisson, binomial, normal, multinomial, and exponential. For an ecological modeler, it is important to get a feel for which statistical distribution is suitable as a description of the randomness in the data for a particular sampling situation, feature of the data collection protocol or measured trait. Experienced modelers therefore have a bestiary of distributions (Bolker, 2008) at their disposal, and they may well try out more than one for the same response to see which is most appropriate. Here, we do not describe these distributions but rather point to other books where some of the key distributions are described in more detail. Examples include Bolker (2008), Royle and Dorazio (2008), Kéry (2010), or Link and Barker (2010).

Program R has a large catalog of distributions that one can use and study to see how each one looks. Check out `?ddist`, and replace `dist` by any of the following: `pois`, `binom`, `norm`, `multinom`, `exp`, and `unif`. Changing the first letter of the function name from `d` to `p`, `q`, or `r` allows one to get the density, the distribution function, the percentiles, and random numbers from these distributions. For instance, to see how a beta distribution with parameters 2 and 4 looks like, you could execute either of the following R commands. Both generate a random sample of size  $n$  from the specified beta distribution and produce a plot. Remember that to know how an R function works you can type a question mark and its name (e.g., `?density`).

```
plot(density(rbeta(n = 10^6, shape1 = 2, shape2 = 4)))
hist(rbeta(10^6, 2, 4), nclass = 100, col = "gray")
```

The GLM in a strict sense has only a single noise component. Often, however, we need several noise components in a statistical model. Such models are introduced in Chapter 4. They include random- or mixed-effects models, hierarchical models, multilevel models, state-space models, or latent-component models. In a sense, they are all fairly similar, although there is a tremendous variety of them. In this book, we usually speak of random-effects or hierarchical models when there is more than a single noise component, except where there is a strong historic precedent favoring one term over the other, such as the state-space models in Chapter 5. In hierarchical models, the definition of a statistical model as being composed of a systematic and a random part must be made separately for each level of the model and what is the random part at one level becomes a component of the systematic part in the next level of the hierarchy.

### 3.2.2 The Signal Component

The signal component of the model contains the predictable parts of a response or the mean structure of a model. One of the most widely used descriptions of the mean structure is by a linear model, although nonlinear models can be adopted as well (Seber and Wild, 2003). The linear model is

one particular way to describe how we imagine that explanatory variables, such as indicators for group memberships or measured covariates, affect an observed response. A linear model is linear in the parameters and does not need to represent a straight line when plotted; most often it does not. This means that the parameters affect the mean response in an additive way, such as  $\alpha$  and  $\beta$  in  $y = \alpha * x_1 + \beta * x_2$ , but not as in  $y = (x_1^\alpha) / (\beta + x_2)$ , where  $x_1$  and  $x_2$  are variables. Most models that ecologists use in their daily work can be represented as linear models: for example, the  $t$ -test, simple and multiple linear regressions, ANOVA, ANCOVA, and mixed models.

All these models can be described in several ways, for example, with a plot, in words or in maths. To be able to code a model in the BUGS language, we need to know how to write it mathematically. Therefore, we need to learn how to describe each model by way of matrices and vectors and, in particular, how to build the so-called design matrix of a model. We briefly illustrate these topics with an ANCOVA linear model for a toy data set of nine data points. We assume that  $y$  is a response,  $A$  is a factor with three levels (i.e., a categorical covariate), and  $x$  is a continuous covariate. Here, they are ready to be defined in R:

```
# Define and plot data
y <- c(25, 14, 68, 79, 64, 139, 49, 119, 111)
A <- factor(c(1, 1, 1, 2, 2, 2, 3, 3, 3))
X <- c(1, 14, 22, 2, 9, 20, 2, 13, 22)
plot(X, y, col = c(rep("red", 3), rep("blue", 3), rep("green", 3)),
     xlim = c(-1, 25), ylim = c(0, 140))
```

In R, we can fit an ANCOVA with parallel slopes by issuing the following command:

```
summary(fm <- lm(y ~ A-1 + X))
[...]
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
A1        1.315    19.260   0.068   0.9482
A2       65.218    17.965   3.630   0.0151 *
A3       58.648    19.260   3.045   0.0286 *
X         2.785     1.031   2.701   0.0427 *
[...]
Residual standard error: 25.05 on 5 degrees of freedom
Multiple R-squared: 0.951,      Adjusted R-squared: 0.9117
F-statistic: 24.24 on 4 and 5 DF, p-value: 0.001798
```

This formula language for defining the linear model underlying the ANCOVA or any other linear model is ingenious because it is quick and error-free *if you know how to specify your model*. The downside is that you must know what this statement actually causes R to do. So, let us look under the hood and see what R does when we tell it  $y \sim A-1 + x$ . When fitting this model, we are in effect fitting the following linear models:

$$y_i = \alpha_{j(i)} + \beta^* X_i + \epsilon_i \quad \text{and} \quad \epsilon_i \sim \text{Normal}(0, \sigma^2)$$

Here,  $y_i$  is the response of unit (data point, individual, row)  $i$ , and  $X_i$  is the value of the continuous explanatory variable  $x$  for unit  $i$ . Factor A codes for the group membership of each unit (for the meaning of the  $-1$ , see below). We have three groups; therefore, the index  $j$  is 1, 2, or 3, and we may write  $j = A_i$ . This means that units with a value  $A_i = 1$  have  $j = 1$  and  $\alpha_1$  and so forth. There are two parameters in the model, which describe the mean structure of the model,  $\alpha$  and  $\beta$ , of which the first is a vector and the second is a scalar. The vector  $\alpha$  has three elements, corresponding to the effects of the three levels of factor A. The part  $\alpha_{j(i)} + \beta^* X_i$  represents the expected response for unit  $i$ , that is, the value expected to be observed in the absence of any noise in the system. Thus, this is the signal part of the response. The noise part consists of that part of the response, which we cannot explain by our linear combination of the explanatory variables: it is represented by the residuals  $\varepsilon_i$ . Since we know a little bit about these noise terms, we claim that they come from a normal distribution with mean equal to zero and common variance  $\sigma^2$ . In all, the model has five parameters:  $\alpha_1$ ,  $\alpha_2$ ,  $\alpha_3$ ,  $\beta$ , and  $\sigma^2$ . A little confusingly, we may also say that the model has three parameters: the vector  $\alpha$  and the scalars  $\beta$  and  $\sigma^2$ .

Other algebraic ways of writing this model clarify perhaps even more the structure of a statistical model as being made up of a signal or systematic part and a noise or random part. The following shows clearly that the response is normally distributed around the values of the linear predictor,  $\alpha_{j(i)} + \beta^* X_i$ . Now, residuals  $\varepsilon_i$  are defined only implicitly.

$$y_i \sim \text{Normal}(\alpha_{j(i)} + \beta^* X_i, \sigma^2).$$

A further possibility is to express  $\mu_i$  as the expected response of unit  $i$  in the absence of any random noise. It is the same as the value of the linear predictor:

$$y_i \sim \text{Normal}(\mu_i, \sigma^2), \text{ with } \mu_i = \alpha_{j(i)} + \beta^* X_i$$

Being able to write a linear model in algebra greatly simplifies the coding of a model in WinBUGS. Most models, when written in the BUGS language, in fact very much resemble their algebraic description.

Yet another way to write this model for our data set is by way of matrices and vectors:

$$\begin{pmatrix} 25 \\ 14 \\ 68 \\ 79 \\ 64 \\ 139 \\ 49 \\ 119 \\ 111 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 14 \\ 1 & 0 & 0 & 22 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 9 \\ 0 & 1 & 0 & 20 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 13 \\ 0 & 0 & 1 & 22 \end{pmatrix} \times \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \\ \varepsilon_9 \end{pmatrix}, \text{ with } \varepsilon_i \sim \text{Normal}(0, \sigma^2)$$

From left to right, we have response vector, design matrix, parameter vector, and residual vector. The design matrix (also called model matrix or X matrix) multiplied with the parameter vector produces another vector, which is the expected value of the response for each unit. This expected value is also called the linear predictor and is the same as  $\mu_i$  above. For example, the value of the linear predictor for the first data point is given by  $1 * \alpha_1 + 0 * \alpha_2 + 0 * \alpha_3 + 1 * \beta$ .

It is useful to practice so that you can jump back and forth between different ways of describing the same model: the R formula language, in algebra, and using matrices and vectors. We need this clear understanding of the linear model if we want to succeed in our modeling with WinBUGS. This may be awkward at first, but a tremendous benefit is that this finally forces on us what we may not have achieved before: a clear understanding of linear models.

The interpretation of the elements of the vector of the mean parameters ( $\alpha_1, \alpha_2, \alpha_3, \beta$ ) depends on the way the design matrix is structured, and there are different ways for doing this. These variations are called parameterizations of a model; they are equivalent ways of writing what is effectively the same model. Sometimes one parameterization may be more convenient and sometimes another. WinBUGS can be very sensitive to the choice of parameterization of a model; sometimes one will not work at all, for example, the chains would not mix, but another will result in beautiful mixing. Hence, it is important that we know how to specify different parameterizations of the same model.

One clever trick when trying to understand the design matrix of a model is the use of the R function `model.matrix()`, especially for those of you who have some experience in fitting linear models with a model-definition language such as that in R. We have had cases in which we did not quite understand the structure of a complicated linear model that we wanted to fit in WinBUGS. To understand the design matrix of the model and, therefore, to be able to specify the model in WinBUGS, we went into R and used `model.matrix()` to find out how the design matrix of the model we wanted to specify in WinBUGS looked like.

As an example, two possible parameterizations of our ANCOVA model are those which we specify by the following R commands:

```
# Effects or treatment contrast parameterization
model.matrix(~ A + X)
  (Intercept) A2 A3  X
1            1  0  0  1
2            1  0  0 14
3            1  0  0 22
4            1  1  0  2
5            1  1  0  9
6            1  1  0 20
7            1  0  1  2
```

```

8          1  0  1  13
9          1  0  1  22
[ ... ]

```

#### # Means parameterization

```
model.matrix(~ A-1 + X)
```

	A1	A2	A3	X
1	1	0	0	1
2	1	0	0	14
3	1	0	0	22
4	0	1	0	2
5	0	1	0	9
6	0	1	0	20
7	0	0	1	2
8	0	0	1	13
9	0	0	1	22
[ ... ]				

The former is the R default called the effects or treatment contrast parameterization. It specifies the linear model in terms of a baseline response, which here is that for the first level of factor  $A$ , plus effects of the other levels of  $A$  relative to the first level and effects of each unit change in the covariate  $x$ . Hence, the intercept is the mean response of units with level 1 of factor  $A$  at  $X=0$ , and the parameters corresponding to the design matrix columns  $A_2$  and  $A_3$  quantify the *difference* in the mean response in levels 2 and 3, relative to that of level 1, for a given value of  $X$ . The parameter representing column  $X$  in the design matrix is the common slope of the regression of  $y$  on  $X$ , regardless of which group a unit belongs to. In contrast, in the means parameterization, the first three parameters directly represent the mean response for each level of factor  $A$  at  $X=0$ , while the meaning of the parameter represented by the column  $X$  is the same as before.

The overly simple setting chosen here allows one to see the main features of these topics very clearly, and in real-world modeling, where one typically has many explanatory variables, one must strive for an understanding of the linear model by breaking down the problem into its smallest understandable parts. In reality, there may be main effects and interaction effects and perhaps aliasing between columns of the design matrix, meaning some parameters cannot be estimated independently. This complicates the construction of the design matrix even more (although most problems are really ones of bookkeeping). We will not go further into these topics, but refer you to any of a huge number of books that explain the linear model, such as Kéry (2010) or chapter 6 in E. Cooch and G. White's free *Gentle Introduction to Program MARK* ([www.phidot.org/software/mark/docs/book](http://www.phidot.org/software/mark/docs/book)).

This concludes our very brief description of how the signal component of a linear statistical model is built using the design matrix, which, when

matrix multiplied with the parameter vector, yields the value of the linear predictor or the expected or “typical” response,  $\mu_i$ , which is also what you get in R by typing `model.matrix(~A+X) %*% fm$coef`.

### 3.2.3 Bringing the Noise and the Signal Components Together: The Link Function

In the linear models mentioned so far in this chapter, we assumed a normal distribution for the random part of the response. Thus, we directly wrote the response  $y_i$  as a simple additive combination of the signal  $\mu_i$  and the noise  $\epsilon_i$ . For responses that cannot be modeled with a normal distribution, this direct combination is no longer possible. For instance, directly adding noise from a Poisson or a binomial distribution to the value of a linear predictor would typically result in inadmissible values for the response, for example, fractional or negative counts.

The big advantage of generalized linear models (GLMs) is that we can apply a linear model to the response indirectly: namely, to a transformation of the mean response. The function that transforms the expected response is called the *link function*. The reason for that name should now be clear: the link function allows us to link the noise component and the signal components in a model. In this way, the useful concepts of linear models can be carried over to a vastly larger class of models.

The classical way to describe a GLM is by three components: a random part (noise), a systematic part (signal), and a link function. More generally, for response  $y_i$ , we can write the following:

1. Random part of the response (the noise)—a statistical distribution  $f$  with mean response  $\mu_i$ :

$$y_i \sim f(\mu_i)$$

2. A link function  $g$ , which is applied to the mean response  $\mu_i$ :

$$g(\mu_i) = \eta_i$$

3. Systematic part of the response (the signal)—a linear predictor ( $\eta_i$ ), for example, for a simple linear regression-type of GLM:

$$\eta_i = \alpha + \beta^* x_i$$

We can describe a GLM succinctly in only two lines:

$$y_i \sim f(\mu_i)$$

$$g(\mu_i) = \alpha + \beta^* x_i$$

This is exactly the way in which GLMs are specified in the BUGS language, and this is the reason why BUGS is so great if you want to really

understand GLMs. The GLM concept gives you considerable creative freedom to combine the three components of a GLM, but there are typically pairs of response distributions and link functions that go together particularly well. These link functions are called canonical link functions and are the identity link for normal responses ( $\eta_i = \mu_i$ ), the log link for Poisson responses ( $\eta_i = \log(\mu_i)$ ), and the logit link for binomial responses ( $\eta_i = \log(\mu_i)/\log(1 - \mu_i)$ ). Together, these three standard GLMs make up a vast number of statistical methods used in population ecology and elsewhere; for an overview, see Kéry (2010).

The broad scope of the GLM is one reason for the great importance of the GLM for you. The other one, which we will see many times later in the book, is that the GLM represents the main building block for many more complicated models, especially hierarchical models. Many of the most exciting ecological models for inference about populations can be viewed simply as a sequence of coupled GLMs (Royle and Dorazio, 2008).

### **3.3 POISSON GLM IN R AND WinBUGS FOR MODELING TIME SERIES OF COUNTS**

The Poisson distribution is defined for positive integers at 0, 1, 2, ... and hence is a suitable model for counts under the assumption of independence and spatial or other randomness. It describes the “residual variation” (noise), after any kinds of systematic effects (signal) in the Poisson mean have been taken account of, for example, in the form of covariate effects.

The Poisson distribution has a single parameter, the intensity or rate parameter  $\lambda$  representing the expected count. The variance of a Poisson random variable is not a free parameter, rather it is identical to the mean (expected) count. In practice, this strong assumption about the mean–variance relationship is frequently violated and the variance of counts is larger than their mean. There are various ways to take account of this. The conceptually easiest is perhaps the introduction of data-level random effects to take account of such overdispersion (Section 4.2).

For an example of a Poisson GLM, let us assume we model counts  $C_i$  from a number of years  $i$ . Here is the algebraic description of the Poisson GLM for count  $C_i$  in year  $i$  with a single covariate  $X$ :

1. Random part of the response (statistical distribution):

$$C_i \sim \text{Poisson}(\lambda_i)$$

2. Link of random and systematic part (log link function):

$$\log(\lambda_i) = \eta_i$$

3. Systematic part of the response (linear predictor  $\eta_i$ ):

$$\eta_i = \alpha + \beta^* X_i$$

Here,  $\lambda_i$  is the expected count (the mean response) in year  $i$  on the arithmetic scale,  $\eta_i$  is the expected count in year  $i$  on the link scale (i.e., the linear predictor),  $X_i$  is the value of covariate  $X$  in year  $i$ , and  $\alpha$  and  $\beta$  are the two parameters of the log-linear regression of these counts on the covariate. We will next look at the generation and analysis of Poisson GLM data for a simulated and also for a real data set.

### 3.3.1 Generation and Analysis of Simulated Data

We define a function that generates Poisson counts of peregrine falcons (Fig. 3.1) for one population over  $n$  years. The parameter values are inspired by actual data from the French Jura mountains (Monneret, 2006), see Section 3.3.2. In this example, the linear predictor will be a cubic polynomial function of time,  $\eta_i = \alpha + \beta_1 * X_i + \beta_2 * X_i^2 + \beta_3 * X_i^3$ . In this section, we will, first, show how a GLM is analyzed in the frequentist framework in R and in the Bayesian framework using WinBUGS and illustrate how similar numerically the resulting estimates are. Second, as this is the first time we run WinBUGS, we explain each step of the analysis in more detail than later in this book.



**FIGURE 3.1** Peregrine falcon (*Falco peregrinus*), Switzerland (Photograph by B. Renevey).

```

data.fn <- function(n = 40, alpha = 3.5576, beta1 = -0.0912,
  beta2 = 0.0091, beta3 = -0.00014) {
  # n: Number of years
  # alpha, beta1, beta2, beta3: coefficients of a
  #   cubic polynomial of count on year

  # Generate values of time covariate
  year <- 1:n

  # Signal: Build up systematic part of the GLM
  log.expected.count <- alpha + beta1 * year + beta2 * year^2 + beta3 *
    year^3
  expected.count <- exp(log.expected.count)

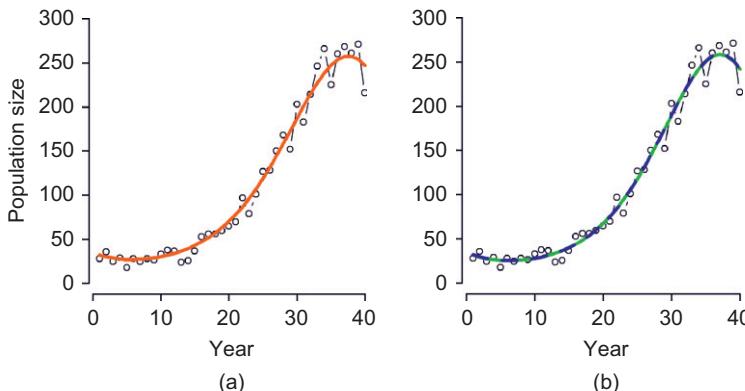
  # Noise: generate random part of the GLM: Poisson noise around
  # expected counts
  C <- rpois(n = n, lambda = expected.count)

  # Plot simulated data
  plot(year, C, type = "b", lwd = 2, col = "black", main = "", las = 1,
    ylab = "Population size", xlab = "Year", cex.lab = 1.2,
    cex.axis = 1.2)
  lines(year, expected.count, type = "l", lwd = 3, col = "red")
  return(list(n = n, alpha = alpha, beta1 = beta1, beta2 = beta2,
    beta3 = beta3, year = year, expected.count = expected.count, C = C))
}

```

We obtain one realization of the stochastic process, that is, population counts over 40 years, and plot the population trajectory over time (Fig. 3.2a)

```
data <- data.fn()
```



**FIGURE 3.2** Simulated population size of peregrines in the French Jura over 40 years. (a) Expected population size (red) and the observed data, the realized population size (black). (b) Observed data (black) and estimated population trajectories from a frequentist (green) and a Bayesian analysis (blue) of a Poisson regression with cubic polynomial effects of year. The R code to produce this figure slightly differs from the one shown in this book.

Next, we analyze this data set in R and in WinBUGS. Fitting a GLM in the frequentist mode of analysis, using the method of maximum likelihood, is trivially easy in statistical software like R that have canned functions such as `glm()`. Here is the analysis using R; one or two lines of code suffice. Up to Poisson sampling error, this will recover parameter estimates that resemble the values of the input.

```
fm <- glm(C ~ year + I(year^2) + I(year^3), family = poisson, data = data)
summary(fm)

Call:
glm(formula = C ~ year + I(year^2) + I(year^3), family = poisson,
     data = data)

Deviance Residuals:
    Min      1Q  Median      3Q      Max 
-1.9036 -0.5815  0.2250  0.8888  1.4972 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 3.570e+00 1.159e-01 30.797 < 2e-16 ***
year        -1.099e-01 2.023e-02 -5.431 5.61e-08 ***
I(year^2)    1.026e-02 1.005e-03 10.204 < 2e-16 ***
I(year^3)   -1.578e-04 1.467e-05 -10.756 < 2e-16 *** 
-
Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 2771.076 on 39 degrees of freedom
Residual deviance: 42.729 on 36 degrees of freedom
AIC: 297.77

Number of Fisher Scoring iterations: 4
```

We note in passing that in order to arrive at this solution, R had to turn the handle on a blackbox called “Fisher Scoring” four times.

In contrast, the code for fitting the same model in WinBUGS is less succinct: about 20 lines compared with 1–2 for the same analysis in R. And this is only for the description of the model in the BUGS language; more lines of code are required to actually conduct the analysis of that model with WinBUGS from R.

```
# Specify model in BUGS language
sink("GLM_Poisson.txt")
cat("
model {

# Priors
alpha ~ dunif(-20, 20)
beta1 ~ dunif(-10, 10)
```

```

beta2 ~ dunif(-10, 10)
beta3 ~ dunif(-10, 10)

# Likelihood: Note key components of a GLM on one line each
for (i in 1:n) {
  C[i] ~ dpois(lambda[i])           # 1. Distribution for random part
  log(lambda[i]) <- log.lambda[i]   # 2. Link function
  log.lambda[i] <- alpha + beta1 * year[i] + beta2 * pow(year[i],2) +
    beta3 * pow(year[i],3)          # 3. Linear predictor
} # i
}
",fill = TRUE)
sink()

```

In this book, we present each WinBUGS analysis in a common layout, by first writing a text file with the model definition in the BUGS language, followed by all the other ingredients that the function `bugs()` in the R2WinBUGS package (Sturtz et al., 2005) requires to instruct WinBUGS from R. Remember that before executing the following code, you must define an R object called `bugs.dir` that contains the address of WinBUGS. For a Swiss-German locale, this might be

```
bugs.dir <- "c:/Programme/WinBUGS14/"
```

Next, we bundle into an R list the data needed for the analysis by WinBUGS.

```

# Bundle data
win.data <- list(C = data$C, n = length(data$C), year = data$year)

```

The next step is to define initial values for the estimated quantities. WinBUGS can generate initial values by drawing them from their priors, so it is not necessary to give inits for all estimands. Nevertheless, when running WinBUGS by calling it from R, we need to define inits for at least one quantity. For complex models, it is often vital to choose good initial values because otherwise WinBUGS may crash. It is useful to define a function to define inits. This function is then executed once for each Markov chain run in the analysis.

```

# Initial values
inits <- function() list(alpha = runif(1, -2, 2), beta1 = runif(1, -3, 3))

```

Next, we write a list with the quantities we want to estimate (“monitor” as WinBUGS calls it), that is, for which we want WinBUGS to save the draws from the joint posterior distribution. This includes derived quantities such as `lambda` here.

```
# Parameters monitored
params <- c("alpha", "beta1", "beta2", "beta3", "lambda")
```

Before running the analysis, we set the MCMC characteristics: the number of draws per chain, thinning rate, burnin length, and the number of chains. The burnin should be long enough to discard the initial part of the Markov chains that have not yet converged to the stationary distribution. We typically determine the required burnin in initial trials. Thinning is useful to save computer disk space. We run multiple chains to check for convergence.

```
# MCMC settings
ni <- 2000
nt <- 2
nb <- 1000
nc <- 3
```

Now, we have defined all R objects that we need as arguments for our call to R function `bugs()`. When calling WinBUGS from R, we usually set the argument `debug = TRUE`. WinBUGS then remains open after the requested number of iterations has been produced, and we can visually inspect whether the chains have converged, or in the case of an error, directly read the log file.

```
# Call WinBUGS from R
out <- bugs(data=win.data, inits=inits, parameters.to.save=params,
    model.file = "GLM_Poisson.txt", n.chains=nc, n.thin=nt,
    n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=bugs.dir,
    working.directory=getwd())
```

Most shockingly, WinBUGS almost immediately crashes, claiming that there was an “undefined real result” trap. Why on Earth should you continue with this book (or even buy the book!) when GLMs can be fitted so very much more easily using functions in software such as R? There are at least three reasons for why you should continue:

1. You can use the same kind of analysis for *much* more complex models, models that you will hardly or not at all be able to fit with R. So, understanding how to fit the simpler models in WinBUGS rather than in R is an essential preparation for that.
2. You conduct a Bayesian analysis instead of a frequentist one. The Bayesian view of statistics has several advantages as argued by many (e.g., McCarthy, 2007; Link and Barker, 2010; also see Chapter 2).
3. The model that you fit is more transparent when using the BUGS language to describe it than when describing it in R. So, there is a huge heuristic benefit of statistical modeling in WinBUGS: that of actually understanding the model you are fitting.

But back to our Bayesian analysis of that really simple model, our Poisson GLM: why was WinBUGS not able to get a solution in this simple case? The answer is that we need to ensure that covariate values are not too far away from zero, that is, that they have neither too large negative nor too large positive values. Note that year<sup>3</sup> goes up to  $40^3 = 64,000$ , and this causes numerical overflow (covariate values in the 10s or even in the 100s should not be a problem). To avoid this problem, we usually center or standardize our covariates. We could use the usual standardization by subtracting the mean and dividing by the standard deviation; this results in transformed covariates with approximately zero mean and unit standard deviation. However, any other transformation usually also works, provided that the range of the transformed covariate does not extend to far on either side of 0. Here, we subtract 20 and divide 10. This is easy and the only cost is when we want to present the results and when we want to compare the input values with our parameter estimates. We will graph the results of the model fitted in WinBUGS with the standardized year covariate. This will convince you that we have actually fitted the equivalent model. So next we repeat the analysis using the standardized covariate values. We can do this simply by adapting the statement in which we package the data and then recycle the rest of the code.

```
# Bundle data
mean.year <- mean(data$year)           # Mean of year covariate
sd.year <- sd(data$year)                # SD of year covariate
win.data <- list(C = data$C, n = length(data$C),
                 year = (data$year - mean.year) / sd.year)

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(data = win.data, inits = inits, parameters.to.save = params,
            model.file = "GLM_Poisson.txt", n.chains = nc, n.thin = nt,
            n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory = bugs.dir,
            working.directory = getwd())
```

This works smoothly and produces a nice first bit of output from our first Bayesian analysis; how reassuring! To return the data into R, you have to manually exit WinBUGS and can then obtain a numerical summary of the Bayesian analysis.

```
# Summarize posteriors
print(out, dig = 3)
Inference for Bugs model at "GLM_Poisson.txt", fit using WinBUGS,
3 chains, each with 2000 iterations (first 1000 discarded), n.thin = 2
n.sims = 1500 iterations saved
      mean    sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
alpha     4.268 0.030   4.208   4.248   4.268   4.288   4.324 1.005   410
beta1    1.308 0.042   1.230   1.277   1.307   1.338   1.391 1.003   590
beta2    0.076 0.024   0.031   0.060   0.076   0.093   0.122 1.013   480
beta3   -0.253 0.022  -0.297  -0.268  -0.253  -0.237  -0.212 1.006   340
```

```

lambda[1] 32.367 3.243 26.579 30.000 32.340 34.492 38.965 1.007 340
lambda[2] 29.815 2.562 25.184 27.997 29.850 31.490 35.057 1.006 390
lambda[3] 27.993 2.068 24.229 26.550 27.995 29.402 32.111 1.005 480
[ ... ]
lambda[38] 257.081 6.451 245.100 252.600 256.900 261.500 269.800 1.001 1500
lambda[39] 251.555 7.723 237.442 246.000 251.300 256.700 267.252 1.000 1500
lambda[40] 242.160 9.271 225.147 235.700 241.900 248.325 261.400 1.000 1500
deviance   293.695 2.622 290.300 291.800 293.200 295.000 300.700 1.001 1500

For each parameter, n.eff is a crude measure of effective sample size, and Rhat is
the potential scale reduction factor (at convergence, Rhat=1).

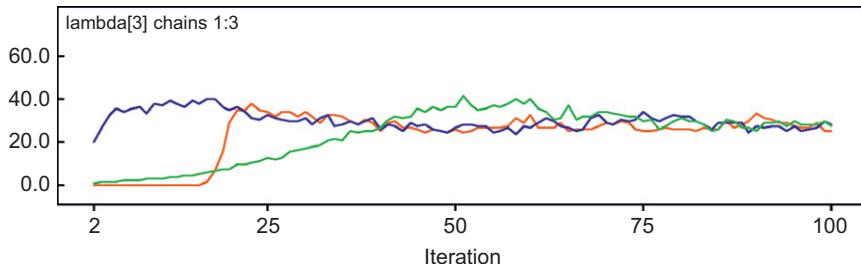
DIC info (using the rule, pD=Dbar-Dhat)
pD = 3.9 and DIC = 297.6
DIC is an estimate of expected predictive error (lower deviance is better).

```

We note that there is a plot function for R objects of the bugs class. Typing `plot(out)` will produce a useful graphical summary of the Bayesian analysis, but we do not show this plot here.

Now, before we even inspect the parameter estimates, we should make sure that their Markov chains have converged. Only then are the random draws produced a valid sample from the desired target distribution, which is the posterior distribution of these parameters. Convergence can never be proven; what *looks* like convergence may indeed sometimes represent chains that have definitely not reached their stationary distribution (see Kéry, 2010, for some nice examples of this). However, in many cases, visual and numerical checks are adequate, and this is what we do in this book: we always inspect the time-series plots in the WinBUGS log file and the values of the Rhat statistics in the table of posterior summaries produced by the function `bugs()`. Rhat is a formal convergence test criterion comparing the among- and the within-chain variance in an ANOVA fashion; values around 1 suggest the absence of a “chain effect” and therefore convergence (Brooks and Gelman, 1998). Often, 1.1 or 1.2 is taken as an Rhat-value that indicates convergence (Gelman and Hill, 2007).

The eye is quite good at picking up a pattern in a graph, and this visual assessment of the likely convergence of chains will complement the numerical assessment by the Rhat values (Fig. 3.3). With experience, you will get a trained eye because you will have seen so many plots of chains that have converged (based on Rhat) and so many others that have not. With `debug = TRUE` as an argument of the function `bugs()`, WinBUGS stays open after execution of the desired number of iterations, and the full WinBUGS functionality can be used for further analyses. For instance, additional iterations could be asked for and numerical summaries produced. However, one may also simply skim over the time-series plots of all monitored parameters to see whether any of them looks like they have not converged (yet). At convergence, the chains should oscillate randomly around a horizontal level and (when three parallel chains are run) should look “grassy” (Link and Barker, 2010). There should not be a sustained trend.



**FIGURE 3.3** Example of a time-series plot with three chains for one parameter (the expected count in year 3). The chains have converged after around 75 iterations and show good mixing. Thus we would repeat the analysis and might set the burnin to at least 75, but probably even to several 100 to be on the safe side.

Furthermore, the lines representing the different chains (usually, 2 or 3) should be strongly interspersed: if this is the case, one says that the chains mix well. In our simple model, we can inspect the time-series plots and see that after the burnin of length 200, the chains of all parameters have indeed converged. We arrive at the same conclusion based on the Rhat values.

To see what initial nonconvergence looks like, you may repeat the analysis with the below MCMC settings. Looking at the plots in WinBUGS shows that convergence is generally achieved after some 75 iterations (Fig. 3.3).

```
# New MCMC settings with essentially no burnin
ni <- 100
nt <- 1
nb <- 1

# Call WinBUGS from R (BRT <1 min)
tmp <- bugs(data=win.data, inits = inits, parameters.to.save = params,
            model.file = "GLM_Poisson.txt", n.chains = nc, n.thin = nt,
            n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory = bugs.dir,
            working.directory = getwd())
```

Once we are satisfied with the convergence of the chains, we can see what we learn from the analysis of the model. For instance, we can plot the Poisson means (the `lambda` parameters) for each year; these represent the expected peregrine counts in each year. We plot them in the same figure as the predicted values from the analysis of the same model using the R function `glm()`, and we will see that our inference is virtually identical (Fig. 3.2b).

```
plot(1:40, data$C, type = "b", lwd = 2, col = "black", main = "", las = 1,
     ylab = "Population size", xlab = "Year")
R.predictions <- predict(glm(C ~ year + I(year^2) + I(year^3),
                             family = poisson, data = data), type = "response")
lines(1:40, R.predictions, type = "l", lwd = 3, col = "green")
WinBUGS.predictions <- out$mean$lambda
lines(1:40, WinBUGS.predictions, type = "l", lwd = 3, col = "blue", lty = 2)
```

We see that the predicted population sizes are so similar from R and WinBUGS that we can hardly see both lines. This is a very common observation: typically, inference from a frequentist and a Bayesian analysis of a statistical model is numerically almost identical when noninformative priors are used. Let us plot the two predictions side by side to convince us that they are indeed so similar:

```
cbind(R.predictions, WinBUGS.predictions)
      R.predictions WinBUGS.predictions
1       32.14482      32.36720
2       29.66780      29.81523
3       27.89637      27.99280
[ .... ]
38      256.76388     257.08067
39      251.21847     251.55480
40      241.79082     242.16020
```

R was able to recover parameter estimates that were very similar to those that we input when assembling the data set. Now we see that the predicted population trajectory from the frequentist analysis in R, using untransformed covariate values, and from the Bayesian analysis in WinBUGS, using transformed covariate values, are virtually identical.

There is always a trade-off between simplicity of model fitting and flexibility: R functions such as `glm()` are very handy, but they can only fit a relatively limited array of models. In addition, and perhaps more importantly, the model fitted with a function like `glm()` is much less transparent than when the same model is fit in WinBUGS. We will see this in the random-effects extension to GLMs in Chapter 4; in WinBUGS, it is conceptually trivial to go from the pure Poisson GLM to the Poisson-lognormal mixed-effects GLM or Poisson generalized linear mixed model (GLMM). In contrast, in R we would have to use a different setting of the same function (`glm(family=quasipoisson)`) or use an altogether different function, such as `lmer()`.

### 3.3.2 Analysis of Real Data Set

We will repeat this analysis now using actual data by analyzing the trajectory of the peregrine population breeding in the French Jura from 1964 to 2003 (R.-J.Monneret, personal communication). Note that by merely conducting this analysis, we implicitly make the assumption that the survey coverage and detection probability of peregrine pairs in the French Jura have not changed in a sustained way over time, otherwise our perceived trends will be distorted (see, e.g., Nichols et al., 2009; Kéry, 2010; Chapter 5). If we have doubts about this important assumption, then a survey design should be used that allows detection probability to be estimated and therefore corrected for. The metapopulation estimation

methods in Chapters 12 and 13 illustrate ways this could be done in the context of population studies of raptors such as the peregrine. Another feature that our model does not address either is possible temporal auto-correlation. To add this, we could use models developed in Chapter 5.

```
# Read data
peregrine <- read.table("falcons.txt", header = TRUE)
```

We attach the data set to be able to directly write the variable names when addressing them.

```
attach(peregrine)
```

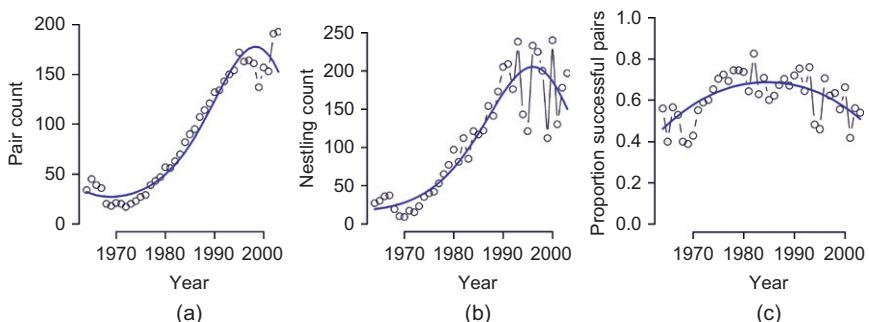
The data set contains the counts of adult pairs (Pairs), reproductive pairs (R.Pairs), and fledged young (Eyasses) for each of 40 years. We plot variable Pairs (Fig. 3.4a).

```
plot(Year, Pairs, type = "b", lwd = 2, main = "", las = 1, ylab = "Pair count",
     xlab = "Year", ylim = c(0, 200), pch = 16)
```

We fit the model in WinBUGS and plot the predictions, vector lambda, into the same plot (Fig. 3.4a, blue line). We will use the same R/WinBUGS code as before.

```
# Bundle data
mean.year <- mean(1:length(Year))           # Mean of year covariate
sd.year <- sd(1:length(Year))                # SD of year covariate
win.data <- list(C = Pairs, n = length(Pairs), year = (1: length(Year)) -
  mean.year) / sd.year

# Initial values
inits <- function() list(alpha = runif(1, -2, 2), beta1 = runif(1, -3, 3))
```



**FIGURE 3.4** Analysis of (a) population size (number of territorial pairs), (b) fecundity (total number of fledged young), and (c) proportion of successful pairs in the peregrine population of the French Jura from 1964 to 2003 (data courtesy of R.-J. Monneret). Observed data are in black and Bayesian posterior means from WinBUGS in blue. The R code to produce this figure slightly differs from the one shown in this book.

```

# Parameters monitored
params <- c("alpha", "beta1", "beta2", "beta3", "lambda")

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out1 <- bugs(data = win.data, inits = inits, parameters.to.save = params,
  model.file = "GLM_Poisson.txt", n.chains = nc, n.thin = nt,
  n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory = bugs.dir,
  working.directory = getwd())

# Summarize posteriors
print(out1, dig = 3)
      mean     sd    2.5%   25%   50%   75% 97.5% Rhat n.eff
alpha    4.234 0.030   4.176   4.214   4.234   4.254  4.293 1.006  400
beta1   1.115 0.047   1.022   1.082   1.116   1.147  1.204 1.009  250
beta2   0.005 0.024  -0.040  -0.012   0.005   0.021  0.052 1.006  350
beta3   -0.233 0.025  -0.280  -0.250  -0.234  -0.215  -0.183 1.010  220
lambda[1] 32.258 3.231  26.499  29.880  32.160  34.450  38.980 1.005  450
lambda[2] 30.221 2.572  25.600  28.367  30.155  31.942  35.550 1.004  570
lambda[3] 28.787 2.088  25.020  27.310  28.710  30.150  33.060 1.003  830
[...]
lambda[38] 169.689 5.294 159.700 166.100 169.700 173.300 179.900 1.001  2800
lambda[39] 162.276 6.248 150.397 158.000 162.200 166.500 174.700 1.002  1500
lambda[40] 152.733 7.347 138.797 147.700 152.600 157.600 167.202 1.003  920
deviance 322.348 2.785 318.900 320.300 321.700 323.600 329.400 1.004  590
[...]
DIC info (using the rule, pD = Dbar - Dhat)
pD = 4.0 and DIC = 326.3

```

Convergence of all chains looks decent; the values of Rhat in the summary are close to 1, and the plots of the chains in WinBUGS look nice. Therefore, we now plot the predicted population trajectory into the previous plot (Fig. 3.4a).

```

WinBUGS.predictions <- out1$mean$lambda
lines(Year, WinBUGS.predictions, type = "l", lwd = 3, col = "blue", lty = 2)

```

## 3.4 POISSON GLM FOR MODELING FECUNDITY

The Poisson distribution is the standard model for any kind of unbounded count data. Counts could be alleles, individuals, family or other groups, or species. To make this very clear, we will swiftly conduct an analogous analysis for the counts of fledged young, that is, for

fecundity, in this same peregrine population (Monneret, 2006). We will use the same model and code and directly plot a Bayesian analysis of a cubic polynomial of the number of fledged young on year (Fig. 3.4b).

```
plot(Year, Eyasses, type = "b", lwd = 2, main = "", las = 1,
     ylab = "Nestling count", xlab = "Year", ylim = c(0, 260), pch = 16)

# Bundle data
mean.year <- mean(1:length(Year))      # Mean of year covariate
sd.year <- sd(1:length(Year))          # SD of year covariate
win.data <- list(C = Eyasses, n = length(Eyasses), year =
  (1: length(Year) - mean.year) / sd.year)

# Call WinBUGS from R (BRT < 1 min)
out2 <- bugs(data = win.data, inits = inits, parameters.to.save =
  params, model.file = "GLM_Poisson.txt", n.chains = nc, n.thin = nt,
  n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory = bugs.dir,
  working.directory = getwd())
```

Skimming over the plots of the Markov chains in WinBUGS and inspecting the values of Rhat in the summary (not shown) suggest that convergence has been reached. Therefore, we are satisfied to plot the estimates under the model (Fig. 3.4b).

```
# Plot predictions
WinBUGS.predictions <- out2$mean$lambda
lines(Year, WinBUGS.predictions, type = "l", lwd = 3, col = "blue")
```

## 3.5 BINOMIAL GLM FOR MODELING BOUNDED COUNTS OR PROPORTIONS

We saw that the Poisson distribution is the standard model for unbounded count data. However, frequently we have counts that are bounded by some upper limit. For instance, when modeling the number of females in a brood, counts of female nestlings cannot exceed the size of a brood. Similarly, when modeling the number of successful broods, counts cannot exceed the total number of broods monitored. As a special case, when modeling the number of survival events for an individual over a single time step, the count cannot exceed 1. The standard model for all these kinds of counts is the binomial distribution. It arises when  $N$  independent individuals all have the same probability  $p$  of experiencing some event (for instance, being female, successful or a survivor). The number of events counted ( $C$ ) will follow a binomial distribution. A special case with  $N=1$  is called a Bernoulli distribution.

As our example for a binomial GLM, we will model the number of successful pairs ( $C_i$ ) among all monitored pairs ( $N_i$ ) in year  $i$  for a total

of 40 years. We will treat year as a continuous covariate and fit a quadratic polynomial. That model can be written like the following:

1. Random part of the response (statistical distribution):

$$C_i \sim \text{Binomial}(N_i, p_i)$$

2. Link of random and systematic part (logit link function):

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \eta_i$$

3. Systematic part of the response (linear predictor  $\eta_i$ ):

$$\eta_i = \alpha + \beta_1 * X_i + \beta_2 * X_i^2$$

Here,  $p_i$  is the expected proportion of successful pairs on the arithmetic scale. It is the mean response for each of the  $N_i$  trials.  $\eta_i$  is that same proportion on the (logit) link scale. The primary parameter of the binomial distribution is  $p_i$ . It is often called the success probability because there are two events: the one of focal interest termed a success and the other a failure. In contrast, the binomial total, or trial or sample size  $N_i$  is not normally a parameter; rather, it is typically observed or a fixed element of the design.

The usual link function adopted for a binomial GLM is the logit. It maps the probability scale (i.e., the range from 0 to 1) onto the entire real line (i.e., from  $-\infty$  to  $\infty$ ) and ensures that a linear model does not result in probabilities outside of that admissible range, that is, below 0 or above 1. The rest of the model (the linear predictor  $\eta_i$ ) is up to your data, your questions, and your imagination. Here,  $\alpha$ ,  $\beta_1$ , and  $\beta_2$  are simply the three parameters of the logit-linear regression of the unobserved proportions on covariate *Year*. Next, we simulate binomial data for the proportion of successful peregrine pairs and analyze them with WinBUGS.

### 3.5.1 Generation and Analysis of Simulated Data

We write a function that simulates data from this simple setting that typically leads to the adoption of a binomial GLM.

```
data.fn <- function(nyears = 40, alpha = 0, beta1 = -0.1, beta2 = -0.9) {
  # nyears: Number of years
  # alpha, beta1, beta2: coefficients

  # Generate untransformed and transformed values of time covariate
  year <- 1:nyears
  YR <- (year-round(nyears/2)) / (nyears / 2)
```

```

# Generate values of binomial totals (N)
N <- round(runif(nyears, min = 20, max = 100))

# Signal: build up systematic part of the GLM
exp.p <- plogis(alpha + beta1 * YR + beta2 * (YR^2))

# Noise: generate random part of the GLM: Binomial noise around
# expected counts (which is N)
C <- rbinom(n = nyears, size = N, prob = exp.p)

# Plot simulated data
plot(year, C/N, type = "b", lwd = 2, col = "black", main = "", las = 1,
     ylab = "Proportion successful pairs", xlab = "Year", ylim = c(0, 1))
points(year, exp.p, type = "l", lwd = 3, col = "red")

return(list(nyears = nyears, alpha = alpha, beta1 = beta1,
            beta2 = beta2, year = year, YR = YR, exp.p = exp.p, C = C, N = N))
}

```

We create one data set, which is inspired by the data in [Section 3.5.2](#).

```
data <- data.fn(nyears = 40, alpha = 1, beta1 = -0.03, beta2 = -0.9)
```

The model as written in the BUGS language is a trivial variant of the Poisson GLM encountered earlier; the key word for the binomial distribution is `dbin()`. Remember that the binomial distribution in WinBUGS is specified with the success parameter ( $p$ ) *before* the binomial total ( $N$ ).

```

# Specify model in BUGS language
sink("GLM_Binomial.txt")
cat("
model {

# Priors
alpha ~ dnorm(0, 0.001)
beta1 ~ dnorm(0, 0.001)
beta2 ~ dnorm(0, 0.001)

# Likelihood
for (i in 1:nyears) {
    C[i] ~ dbin(p[i], N[i])      # 1. Distribution for random part
    logit(p[i]) <- alpha + beta1 * year[i] + beta2 * pow(year[i], 2) # Link
                           # function and linear predictor
}
", fill = TRUE)
sink()

```

We choose a different scaling of `year` this time and simply subtract 20 and divide the result by 20. This leads to values of the covariate that range from about  $-1$  to  $1$ .

```

# Bundle data
win.data <- list(C = data$C, N = data$N, nyears = length(data$C) ,
                  year = data$YR)

# Initial values
inits <- function() list(alpha = runif(1, -1, 1), beta1 =
                           runif(1, -1, 1), beta2 = runif(1, -1, 1))

# Parameters monitored
params <- c("alpha", "beta1", "beta2", "p")

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(data = win.data, inits = inits, parameters.to.save = params,
            model.file = "GLM_Binomial.txt", n.chains = nc, n.thin = nt,
            n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory = bugs.dir,
            working.directory = getwd())

```

We first check convergence by looking at the plots in WinBUGS and skimming over the values in the column entitled Rhat in the summary of the analysis in R (not shown). Both tests look very satisfactory, so we add a plot of the predicted proportion of successful pairs.

```

# Plot predictions
WinBUGS.predictions <- out$mean$p
lines(1:length(data$C), WinBUGS.predictions, type = "l", lwd = 3,
      col = "blue", lty = 2)

```

### 3.5.2 Analysis of Real Data Set

We fit the same model to the proportion of successful peregrine pairs in the French Jura. We do not need to define the model again, since we did this in the previous section already.

```

# Read data and attach them
peregrine <- read.table("falcons.txt", header = TRUE)
attach(peregrine)

# Bundle data (note yet another standardization for year)
win.data <- list(C = R.Pairs, N = Pairs, nyears = length(Pairs),
                  year = (Year - 1985) / 20)

# Initial values
inits <- function() list(alpha = runif(1, -1, 1), beta1 = runif(1, -1, 1),
                           beta2 = runif(1, -1, 1))

# Parameters monitored
params <- c("alpha", "beta1", "beta2", "p")

```

```

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out3 <- bugs(data=win.data, inits=inits, parameters.to.save =
  params, model.file = "GLM_Binomial.txt", n.chains=nc, n.thin=nt,
  n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=bugs.dir,
  working.directory=getwd())

# Summarize posteriors and plot estimates
print(out3, dig=3)
plot(Year, R.Pairs/Pairs, type = "b", lwd = 2, col = "black", main = "",
  las = 1, ylab = "Proportion successful pairs", xlab = "Year",
  ylim=c(0,1))
lines(Year, out3$mean$p, type = "l", lwd = 3, col = "blue")

```

Convergence looks wonderful: the plots in WinBUGS look “grassy” (Link and Barker, 2010) and all Rhat values are close to 1. The proportion of successful pairs increased and then declined again over the years (Fig. 3.4c). The initial increase may be due to the recovery from pesticide effects and the decline to density dependence and the spread of a predator, the eagle owl.

## 3.6 SUMMARY AND OUTLOOK

In this chapter, we have covered much and important ground. We have illustrated the concept of a statistical model as being composed of a statistical distribution, to account of the noise, and a linear predictor for the signal. We have also introduced generalized linear models (GLMs); they allow distributions other than the normal to model the noise in a response. GLMs do so by using a link function, a transformation of the mean response that linearizes the relationship between the transformed mean response and covariates. We examined two common GLMs: Poisson and binomial. With the Poisson GLM, we saw how a classical analysis using maximum likelihood typically yields estimates that numerically closely match those from a Bayesian analysis with vague priors. In every example, we have illustrated convergence assessment of Markov chains by visual and formal means. We have modeled the temporal variation in counts exclusively, but the Poisson and binomial distributions may also be used to describe spatial variation of counts. We expect that much of this is a repetition for you rather than the first time you encounter the concept of a GLM. Otherwise it will be beneficial to first read more specific references that deal with this essential topic of applied statistical

modeling, for example, Crawley (2005), Dobson and Barnett (2008), and Kéry (2010). In the next chapter, we will generalize the GLM to include so-called random effects and to become a generalized linear mixed model (GLMM), an example of a hierarchical model. One other interesting extension of the GLM, to include nonlinear, “wiggly” terms to become a generalized additive model (GAM; Hastie and Tibshirani, 1990), is not dealt with in this book. However, such spline models can be implemented in WinBUGS as sort of random-effects model. For code examples, see Gimenez et al. (2006a, b) and Grosbois et al. (2009).

### 3.7 EXERCISES

---

1. Adapt the first data-generation function in this chapter to generate the data using coefficients that refer to the values of standardized covariate values and repeat the analysis in R and in WinBUGS.
2. Take the following toy data set and fit a logistic regression of the number of successes  $r$  among  $n$  trials as a function of covariate X. Also write out the GLM for this data set.

```
n <- c(22, 8, 10, 7, 10, 6, 11)
r <- c(20, 7, 10, 6, 0, 1, 2)
x <- c(0, 3, 1, 4, 5, 8, 10)
```

3. The Bernoulli distribution is a special case of the binomial with trial size equal to 1. It has only one parameter, the success probability  $p$ . The Bernoulli distribution is a conventional model for species distributions, where observed detection or nondetection data are related to explanatory (e.g., habitat) variables in a linear or other fashion with a logit link. Write an R function to assemble “presence or absence” data collected at 200 sites, where the success probability (i.e., occurrence probability) is related to habitat variable X (ranging from -1 to 1) on the logit-linear scale with intercept -2 and slope 5. Then write a WinBUGS program to “break down” the simulated data (i.e., analyze them) and thus recover these parameter values.
4. In Section 3.5.2, we used a binomial GLM to describe the proportion of successful peregrine pairs per year in the French Jura mountains. To see the connections between three important types of GLMs, first use a Poisson GLM to model the number of successful pairs (thus disregarding the fact that the binomial total varies by year), and second, use a normal GLM to do the same. In the same graph, compare the predicted numbers of successful pairs for every year under all three models (binomial, Poisson, and normal GLMs). Do this in both R and WinBUGS.

# Introduction to Random Effects: Conventional Poisson GLMM for Count Data

## OUTLINE

<b>4.1</b>	<b>Introduction</b>	<b>73</b>
4.1.1	<i>An Example</i>	74
4.1.2	<i>What Are Random Effects?</i>	77
4.1.3	<i>Why Do We Treat Batches of Effects as Random?</i>	78
4.1.4	<i>Why Should We Ever Treat a Factor as Fixed?</i>	82
<b>4.2</b>	<b>Accounting for Overdispersion by Random Effects-Modeling in R and WinBUGS</b>	<b>82</b>
4.2.1	<i>Generation and Analysis of Simulated Data</i>	84
4.2.2	<i>Analysis of Real Data</i>	88
<b>4.3</b>	<b>Mixed Models with Random Effects for Variability among Groups (Site and Year Effects)</b>	<b>90</b>
4.3.1	<i>Generation and Analysis of Simulated Data</i>	92
4.3.2	<i>Analysis of Real Data Set</i>	95
<b>4.4</b>	<b>Summary and Outlook</b>	<b>110</b>
<b>4.5</b>	<b>Exercises</b>	<b>112</b>

## 4.1 INTRODUCTION

In Chapter 3, we have seen that the generalized linear model (GLM) is an extremely flexible and useful model and that much of applied statistics in ecology is based on GLMs. Now, we introduce perhaps the most

important extension of the GLM: random effects. A GLM with fixed and random effects is also called a generalized linear mixed model (GLMM; Bolker, 2008). In this section, we explain what random effects are and describe the reasons for wanting to describe, and the consequences of declaring, a set of effects as random. After that, we look at a simple example of a Poisson GLMM, where random year effects are used to account for overdispersion in a time series of counts (Section 4.2). The resulting Poisson–log-normal model may not be the most familiar application of random effects, but it builds directly on the peregrine GLM example in Chapter 3. Furthermore, it illustrates the important topic of accounting for overdispersion in nonnormal GLMs for count data (Lee and Nelder, 2000; Millar, 2009). In Section 4.3, we fit more complex GLMMs with multiple sets of random effects. This is the kind of random effects that you may be more familiar with: latent, that is, unobserved, effects that are shared by all members of a group and that induce a correlated response among members of the same group. We call all models in this chapter “conventional” mixed models because they are not based on any underlying population model such as the mixed models in Chapter 5.

### 4.1.1 An Example

To illustrate, we revisit the ANCOVA example from Section 3.2.2, simply with changed variable names. We assume that we studied the mass–length relationship in *asp* vipers (Fig. 1.4) and had examined nine individuals in three populations. Here are the data and code for a plot.

```
# Define and plot data
mass <- c(25, 14, 68, 79, 64, 139, 49, 119, 111)
pop <- factor(c(1, 1, 1, 2, 2, 2, 3, 3, 3))
length <- c(1, 14, 22, 2, 9, 20, 2, 13, 22)
plot(length, mass, col = c(rep("red", 3), rep("blue", 3),
  rep("green", 3)), xlim = c(-1, 25), ylim = c(0, 140), cex = 1.5, lwd = 2,
  frame.plot = FALSE, las = 1, pch = 16, xlab = "Length", ylab = "Mass")
```

The plot suggests a linear relationship between mass and length with a different baseline in each population. A regression model ought to account for population differences. In one of the simplest such models, the mass of snake  $i$  depends on length in a linear way, allows populations  $j$  to have a different intercept, and has residuals  $\varepsilon_i$  come from a zero-mean normal distribution with variance  $\sigma^2$ .

$$\text{mass}_i = \alpha_{j(i)} + \beta * \text{length}_i + \varepsilon_i$$

$$\varepsilon_i \sim \text{Normal}(0, \sigma^2)$$

One way to motivate random effects is to recognize that there are two assumptions that one could make about the three asp viper populations studied:

1. They are the only populations that we are interested in.
2. They merely represent a sample from a larger number of populations that we *could* have studied and we would like to generalize our conclusions to this larger statistical population of snake populations.

These assumptions about the populations can be translated into the following assumptions about the population effects  $\alpha_j$ :

1. The three  $\alpha_j$  are completely independent.
2. The three  $\alpha_j$  are not independent; instead, we regard them as a sample from a larger number of effects, which *could* have appeared in our study, and we would like to learn something about that population of effects.

Traditionally, assumption 1 leads one to treat the  $\alpha_j$  as fixed effects, while assumption 2 leads to the adoption of a random-effects model for the effects  $\alpha_j$ . Algebraically, the *only* difference between the two models is that for the random-effects ANCOVA, we add a distributional assumption about the intercepts of the three regression lines, that is, about the population effects  $\alpha_j$ .

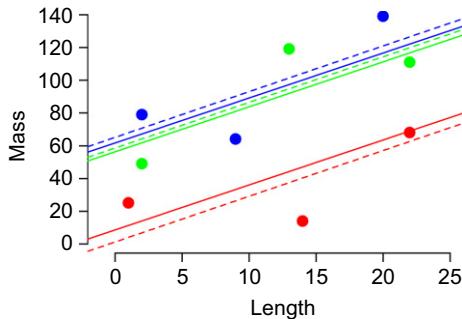
$$\text{mass}_i = \alpha_{j(i)} + \beta * \text{length}_i + \varepsilon_i$$

$$\varepsilon_i \sim \text{Normal}(0, \sigma^2)$$

$$\alpha_j \sim \text{Normal}(\mu_\alpha, \sigma_\alpha^2). \quad \text{This line makes } \alpha_j \text{ random!}$$

The third equation is the *only* difference between a model that treats the population effects  $\alpha_j$  as fixed and one where  $\alpha_j$  are random: without that line, the effects are fixed, and with it, they are random. Line 3 defines  $\alpha_j$  as random draws from a normal distribution with mean  $\mu_\alpha$  and variance  $\sigma_\alpha^2$ . This model is also called a random-intercepts model and is an example of a mixed model, that is, one with both random effects ( $\alpha_j$ ) and fixed effects ( $\beta$ ). See Kéry (2010) for many further examples of fixed-effects models and their random-effects analogues; also see [Section 4.3.2](#).

We use R to fit both models and plot the resulting regression lines ([Fig. 4.1](#)). For the random-effects model, we use the REML method, a variant of maximum likelihood that is better suited for mixed models and fit the model with the function `lmer` in the R package `lme4`, which we need to load first.



**FIGURE 4.1** Relationship between mass and length of asp vipers in three populations (indicated by colors). Circles are the nine measurements. Dashed lines are the estimated regressions for each population under an ANCOVA model with fixed population effects (intercepts). Solid lines are the estimated regressions under a mixed ANCOVA model with random population effects (intercepts). Intercepts under the random-effects ANCOVA are shrunk toward the grand mean, that is, the average intercept.

```
# Fit fixed-effects model, print regression parameter estimates and
# plot regression lines
summary(lm <- lm(mass ~ pop-1 + length))
abline(lm$coef[1], lm$coef[4], col = "red", lwd = 3, lty = 2)
abline(lm$coef[2], lm$coef[4], col = "blue", lwd = 3, lty = 2)
abline(lm$coef[3], lm$coef[4], col = "green", lwd = 3, lty = 2)

# Fit mixed model, print random effects and plot regression lines
summary(lmm <- lmer(mass ~ length + (1|pop)))
ranef(lmm)
abline((lmm@fixef[1]+ranef(lmm)$pop)[1,], lmm@fixef[2], col = "red",
       lwd = 3)
abline((lmm@fixef[1]+ranef(lmm)$pop)[2,], lmm@fixef[2], col = "blue",
       lwd = 3)
abline((lmm@fixef[1]+ranef(lmm)$pop)[3,], lmm@fixef[2], col = "green",
       lwd = 3)
```

Thus, when we describe a model in algebra, the difference between a fixed- and the random-effects version of a model is really trivial. Nevertheless, random effects appear to be a fairly challenging concept for many ecologists. This may have to do with two things. First, random effects are often introduced within the context of ANOVA decompositions of sums of squares. This may simply not be the best setting within which to explain the concepts of the actual models. Second, in almost all software, there is a big divide between procedures that are used to fit the fixed- and the random-effects versions of a model. For instance, in R, there is `lm` for fixed-effects normal linear models and `lmer` for random-effects normal linear models. The two have quite a different syntax and `lmer` cannot even be used to fit fixed-effects models, and vice versa. Hence, the differences rather than the similarities between the two versions of a model are emphasized.

Fortunately, this is different with WinBUGS and the BUGS language; here, we can easily switch between fixed and random effects in a model. We have met quite a few ecologists who have only started to understand random effects after fitting random-effects models in WinBUGS. In the next sections, we try to explain in more detail random effects, the motivation for treating a set of effects as random, and why we will not always treat all effects in a model as random. When possible, we will refer back to the introductory example.

### 4.1.2 What Are Random Effects?

Put simply, random effects are two or more effects or parameters that “belong together” in some way. Specifically, they are estimated under the constraint of a common distribution. We may imagine that a common stochastic process has generated the effects. Therefore, replication of a study would yield a different set of realizations from that random process, that is, a different sample of effects from the distribution describing the stochastic process. In the Bayesian literature, one often reads that to assume effects as random, they must be exchangeable. Exchangeability implies similar, but not identical effects, and is a judgment, not something that can be tested. We are usually free to choose to treat a factor as fixed or random, but there are cases in which the assumption of exchangeability (treating effects as random) would not be biologically justified. For instance, if two of our snake populations are wild and one living in captivity, we would probably not choose to treat the intercepts as random. Rather, we would do so only after having accounted for another fixed effect coding for life in the wild versus in captivity.

The distribution that collects together a set of random effects is called a prior distribution by Bayesians. This may be slightly confusing, since it is not exactly the same kind of prior distribution as the distribution with which we describe our knowledge about each parameter of a model; this typically has known constants only. Instead, a prior governing a set of random effects itself has parameters that must be estimated and which therefore need priors themselves. Hence, the parameters of the random-effects distribution become hyperparameters and their priors become hyperpriors. Such a hierarchical scheme may include additional levels, so we might have hyper-hyperparameters and so forth. Of course, these names would soon become unwieldy, and indeed, are uninteresting, but it is important to recognize the common principle: as soon as we assume effects are random, we introduce a hierarchy of effects. The parameters at one level of the hierarchy can be seen as the realization of some probability distribution, the parameters of which may themselves be the realization of another probability distribution one level up. Hence, any model with random effects is intrinsically a hierarchical model; see Section 2.8.

Typically in ecology, random effects are continuous, and a normal distribution is assumed for them. However, random effects may also be discrete, for instance, in the state-space representation of survival models (Chapters 7–11), in binomial mixture models (Chapter 12), or in occupancy models (Chapter 13). The prior distribution for these discrete random effects is the Bernoulli and the Poisson.

We find the terminology surrounding random-effects models fairly confusing, so here we try to make some connections:

- Random effects always imply grouped effects; we model grouped parameters, batches, or sets of parameters (Gelman, 2005). We cannot declare a single effect as random.
- The factor that indexes group membership is called a random-effects factor or random factor for short, and the groups of a factor in general (fixed or random) are also called levels.
- The simplest random effects perhaps are intercepts or mean effects, and the associated models are sometimes called random-intercepts or varying-intercepts models (Gelman and Hill, 2007). However, we can just as well assume that a collection of slopes comes from a common distribution, leading to a random-coefficients or varying-slopes model (Gelman and Hill, 2007). Typically, random-coefficients models also contain random intercepts.
- We can also model variance parameters as random (Lee and Nelder, 2006).
- Models with random-effects factors are typically called random-effects models. When they also have fixed effects, they become mixed models or mixed-effects models. The following names are largely synonymous or at least overlap with the term, mixed model: variance-components model, hierarchical model, state-space model, latent-component, and latent-variable model.

#### 4.1.3 Why Do We Treat Batches of Effects as Random?

There are many different motivations for random-effects modeling. This section lists some of them.

##### ***Scope of Inference***

With random effects, the scope of inference of a study extends beyond the particular levels of a factor that appear in the study. In our snake example, this formally allows us to generalize to a larger number of populations, from which the three population studied were drawn from. Hence, we buy greater generality in treating our populations as representatives of a larger (statistical) population of populations that we *could* have studied and about which we would also like to learn something, for instance about how they

vary among each other. In truth, such a population of populations may be fairly hypothetical and hard to define in practice, yet it may be sensible to assume it exists. Of course, we can estimate the effects of the particular populations in our study. In addition, when random, we can also estimate or predict the effects of populations that are *not* in the studied sample but that belong to the same statistical population of populations. This is interesting for predicting a process into the future, for instance, when temporal random effects are included in a model, or for making similar extrapolations over space with spatial random effects.

### **Assessment of Variability**

A random-effects model often focuses on the variability in a process. For instance, in a population viability analysis (Beissinger, 2002), the appropriate way to estimate temporal variance in a quantity like a survival probability is to model annual survival as a random effect (see Chapter 7). Actually, we can then estimate two kinds of variability: first, the variation among the levels in the population, that is, the population standard deviation (or variance); this is the parameter estimated in the model (e.g.,  $\sigma_a^2$  in the snake example). However, we can also estimate the variation among the observed levels in our study, that is,  $sd(\alpha_j)$  in our example. This is called the finite-population (or finite-sample) standard deviation. Gelman (2005) suggests the latter as a unified measure of the importance of a factor.

### **Partitioning of Variability**

A random-effects factor codes for unstructured variability among the units in a group. For instance, it quantifies the variability in some process such as survival over a series of years. We may then be interested to explain that variability by covariates, for instance, a weather covariate measured every year. Covariate effects can be assessed by fitting a model first with a random-effects factor alone, for example, time, and second with a temporally varying covariate included. The proportional reduction in the time variance component is a measure for the proportion of the temporal variance in survival explained by that covariate (e.g., Grosbois et al., 2008; see also Section 7.4.3).

### **Modeling of Correlations among Parameters**

Since we can assess the variation among the levels of one factor using random effects, we can also assess the covariation among pairs of levels of two factors using correlated random effects. For instance, we may model a correlation between intercepts and slopes (see Kéry, 2010) or between pairs of annual values of survival and fecundity (Cam et al., 2002) or juvenile and adult survival (see Section 7.6.2). Similarly, most modeling of spatial or temporal autocorrelation first declares random site or time effects, on which a correlation structure can then be imposed. For instance, a random

time effect at  $t + 1$  may be assumed to depend on the time effect at  $t$  in an autoregressive time series (Chapter 5). Similarly, spatial correlation may be modeled by assuming a correlation matrix for random site effects, with the correlation being a function of the distance between the sites (e.g., Royle et al., 2007b).

### ***Accounting for All Random Processes in a Modeled System***

Random-effects modeling results in a more honest accounting for the total uncertainty in a modeled system. Treating some components as random rather than as fixed recognizes the additional variability in, and hence uncertainty stemming from, these components. For instance, in our snake example, a replication of the study might have chosen different populations, with slightly different average mass, and by treating snake populations as random rather than fixed accounts for the fact that this sampling process introduces additional variability in our study, that is, additional uncertainty in our conclusions. If we estimate mass-length regression lines without accounting for this component of variation among populations and implicitly assume that our results are valid for other snake populations as well, our standard error for the regression lines will be too small. Thus, random effects account for the added uncertainty in the form of the sampling error incurred by sampling just some populations in a study, while many other, different, but similar and therefore exchangeable populations could have been selected instead.

### ***Avoiding Pseudoreplication***

Historically, one of the first motivations for random-effects modeling was the desire to account for inherent structure in many data sets, that is, to correct for intrinsic correlations and thereby avoid pseudoreplication (Hurlbert, 1984). For instance, in a study where plant height is measured for each of a sample of plants drawn from multiple populations, a random population effect will avoid committing pseudoreplication. The shared population effect in the measurement of two plants in the same population induces a correlation in their height. This correlation coefficient is sometimes called intraclass correlation; it is equal to  $\sigma_{\text{pop}}^2 / (\sigma_{\text{pop}}^2 + \sigma_{\text{res}}^2)$ , where  $\sigma_{\text{pop}}^2$  is the population variance component, and  $\sigma_{\text{res}}^2$  the residual variance component among plants within populations. Similarly, the blocking structure in a designed experiment will be accounted for by random block effects.

### ***Borrowing Strength***

Random-effects modeling consists of constraining grouped parameters by a common distribution. Effects are no longer estimated independently from one another; rather, the estimate of each effect is influenced by all

members in the group. Thus, the groups share information. One also says that individual estimates “borrow strength from the ensemble”. If the assumption of exchangeability holds, individual estimates will be improved (Link and Sauer, 1996; Sauer and Link, 2002; Welham et al., 2004). One consequence is that individual estimates are pulled in (shrunk) toward the grand mean, an effect called shrinkage. Effects that are estimated with less precision or are more extreme are pulled in more than the effects estimated with greater precision and that are more “average”. This can be seen in the snake example, where the intercept of the more extreme red population is shrunk most. Shrinkage is greater when group effects ( $\sigma_{\text{pop}}^2$ ) are smaller relative to the residual variation ( $\sigma_{\text{res}}^2$ ), that is, when the intraclass correlation is small. With large sample size in each group (i.e., small  $\sigma_{\text{res}}^2$ ) and highly variable groups (large  $\sigma_{\text{pop}}^2$ ), hardly any shrinkage takes place. Random-effects estimates then become essentially identical to fixed-effects estimates. In fact, fixed-effects are a special case of random-effects estimates when the group variance component ( $\sigma_{\text{pop}}^2$ ) is infinite.

### ***Random Effects as a Compromise between Pooling and No Pooling of Batched Effects***

Random-effects modeling can also be seen as an alternative to model selection: rather than making a hard decision about whether a factor should be in the model or out, we let the data (and the model) decide and allow a factor to be in the model in a fractional manner. Gelman and Hill (2007) argue that the random-effects assumption represents an intermediate between assuming all groups (levels) of a factor have an effect and assuming that none of them has an effect. Under the first assumption, all effects are independent and not pooled; this results in the classical fixed-effects estimates. Under the second assumption, the effects are absent or pooled, that is, we simply estimate the grand mean. In contrast, the random-effects assumption is equivalent to partial pooling of the effects, with the degree of pooling depending on the relative size of the among-group variation ( $\sigma_{\text{pop}}^2$ ) and the within-group variation ( $\sigma_{\text{res}}^2$ ) and on the precision of each individual group mean estimate.

### ***Combining Information***

Random-effects modeling is a natural way in which information may be combined over groups. For instance, results from different studies can be combined by assuming study random effects in a meta-analysis. The combined estimate is then automatically weighed by the information content of each study. Similarly, when combining the analysis of multiple data sets, each data set might be considered a sample from some larger hypothetical population of such data sets and modeled as a random effect.

#### 4.1.4 Why Should We Ever Treat a Factor as Fixed?

So why do we not always treat all effects as random? There are several reasons.

First, the assumption of exchangeability may simply not hold and units in a group of effects may differ systematically. The assumption of a common prior distribution is then not sensible. “Borrowing strength” would backfire in these cases, by pulling extreme effects estimates toward the mean of the effects, in spite of them being *really* different and extreme.

Second, treating a factor with very few levels as random will result in very imprecise estimates of the hyperparameter. Hence, when factors have fewer than, say, 5–10 levels, they will rarely be treated as random (but see Gelman, 2005).

Third, random effects are computationally more expensive. We will see this many times with WinBUGS: as soon as we move from a fixed-effects formulation of a model to the corresponding random-effects version of the model, we need to run longer Markov chains to get convergence and run times will be increased, and sometimes greatly so. Similarly, it is often more difficult to get converging Markov chains in the analysis of random-effects versions of a model compared with the grouped effects when they are assumed fixed.

Fourth, another practical reason may be that many ecologists find random-effects models more difficult to understand. We believe that this has to do with the way in which random-effects models are presented in statistics classes at university, that is, within an ANOVA framework. To many, this is challenging and somewhat opaque. In contrast, within a linear model framework, with the models written in algebra, it becomes much more transparent what random effects are, as we have seen above. Another practical reason may be that historically, flexible software for modeling random effects has become widely available much later than software for modeling traditional fixed-effects models.

## 4.2 ACCOUNTING FOR OVERDISPERSION BY RANDOM EFFECTS-MODELING IN R AND WinBUGS

We introduce random effects in an application that may not be so familiar: as a way of accounting for overdispersion in count data. Overdispersion in Poisson or binomial responses denotes the situation that a response is more variable than prescribed by these two distributions where the variance is a function of the mean and thus not a free parameter. Overdispersion arises when important covariates are not in the model or when units are not independent, such as when nestlings are grouped in the same nests or plants within the same population.

Not accounting for this overdispersion would lead to too liberal tests and too short (e.g., 95% credible) intervals. In our example, we estimate one random effect for each observation. Thus, in a sense, we fit extra residuals in addition to the implicit residuals coming with the Poisson assumption and assume that the extra residuals are normal on the log link scale. More typical applications of random-effects modeling usually have effects that are shared by more than one observation (see Section 4.3 for random site and random year effects).

Overdispersion is very frequent when modeling counts; actually, it is the rule rather than the exception. The variance of counts modeled by a Poisson is not a free parameter, so when fitting a Poisson GLM with a function such as `glm()`, the residual deviance of a model should be about the same as the residual degrees of freedom. For example, look at the analysis of the (real) peregrine counts in Section 3.2.2:

```
[ ... ]
Null deviance: 1591.395 on 39 degrees of freedom
Residual deviance: 76.448 on 36 degrees of freedom
[ ... ]
```

The residual deviance is more than twice as large as the residual degrees of freedom, a clear indication that the observations (i.e., the counts) are more variable than expected under the Poisson assumption. Since we have one observation per year, we may imagine the presence of unmodeled year effects. These can be modeled as coming from a normal distribution. The resulting model is also called the Poisson–log-normal (or PLN) model; see also chapter 15.1 in Gelman and Hill (2007), or Millar (2009). Other possibilities to account for overdispersion in R would be to use a negative binomial distribution or quasi-likelihood (Lee and Nelder, 2000). For the latter, see the family argument `quasipoisson` in the R function `glm()`.

As usual, it is enlightening to write down the model in algebra. In Section 3.2.2, we had assumed the following GLM for count  $C_i$  in year  $i$ :

1. Statistical distribution to describe the random part of the response:  
Poisson

$$C_i \sim \text{Poisson}(\lambda_i)$$

2. Link function to transform the mean of the parameter so that it can be expressed as a linear function of covariates and random effects: the log

$$\log(\lambda_i) = \eta_i$$

3. Linear predictor to describe the systematic part of the response:

$$\eta_i = \alpha + \beta_1 * \text{year}_i + \beta_2 * \text{year}_i^2 + \beta_3 * \text{year}_i^3 \quad (\text{old linear predictor})$$

So far, the model is the usual Poisson GLM. To convert it into a Poisson GLMM, we simply introduce into the linear predictor random effects  $\epsilon_i$ . Hence, we replace component 3 with a new linear predictor:

$$\eta_i = \alpha + \beta_1 * \text{year}_i + \beta_2 * \text{year}_i^2 + \beta_3 * \text{year}_i^3 + \epsilon_i \quad (\text{new linear predictor})$$

The fourth component of the model is the distribution of the extra residuals:

4. Distribution of extra residuals ( $\epsilon_i$ ): Normal

$$\epsilon_i \sim \text{Normal}(0, \sigma^2)$$

We could have written the same model without the  $\epsilon$  terms, but instead making the intercepts year specific and collecting them together in a normal distribution like the following:  $\alpha_i \sim \text{Normal}(\mu, \sigma^2)$ . This is simply a reparameterization of the same model.

This completes our algebraic description of the model, and we will recognize later how the model description in the BUGS language is almost a direct transcription of this algebraic description. Note that the random effects  $\epsilon$  are year specific, so we may call them random year effects, but we should not confuse them with the three polynomial regression effects of year. Also, we may only think of overdispersion here as being caused by unmodeled year effects because we have a single observation per year.

#### 4.2.1 Generation and Analysis of Simulated Data

We will now assemble random-effects counts and then break them down using mixed modeling in R and WinBUGS. This will illustrate again that inference from frequentist and Bayesian analyses will often be numerically very similar. Note that some versions of the lme4 package do not allow the fitting of the PLN model, so you will have to choose a version of the package that does (e.g., 2.8.1 or 2.12.1).

```
data.fn <- function(n = 40, alpha = 3.5576, beta1 = -0.0912, beta2 = 0.0091,
  beta3 = -0.00014, sd = 0.1) {
  # n: Number of years
  # alpha, beta1, beta2, beta3: coefficients of a
  #   cubic polynomial of count on year
  # sd: standard deviation of normal distribution assumed for year
  #   effects

  # Generate values of time covariate
  year <- 1:n

  # First level of noise: generate random year effects
  eps <- rnorm(n = n, mean = 0, sd = sd)
```

```

# Signal (plus first level of noise): build up systematic part of the
# GLM and add the random year effects
log.expected.count <- alpha + beta1 * year + beta2 * year^2 + beta3 *
    year^3 + eps
expected.count <- exp(log.expected.count)

# Second level of noise: generate random part of the GLM: Poisson
# noise around expected counts
C <- rpois(n = n, lambda = expected.count)

# Plot simulated data
plot(year, C, type = "b", lwd = 2, main = "", las = 1, ylab = "Population
    size", xlab = "Year", ylim = c(0, 1.1*max(C)))
lines(year, expected.count, type = "l", lwd = 3, col = "red")

return(list(n = n, alpha = alpha, beta1 = beta1, beta2 = beta2, beta3 =
    beta3, year = year, sd = sd, expected.count = expected.count, C = C))
}

```

We generate one data set. The expected count now contains a random contribution from each year, so it is no longer a smooth line as in Chapter 3. We can also say that the response contains two random components now: one from the year and the other a common Poisson residual.

```
data <- data.fn()
```

In R, we use the function `lmer()` in the `lme4` package to fit the model.

```

library(lme4)
yr <- factor(data$year)           # Create a factor year
glmm.fit <- lmer(C ~ (1 | yr) + year + I(year^2) + I(year^3), family =
    poisson, data = data)

Warning message:
In mer_finalize(ans) : false convergence (8)

```

We fail to get convergence, so we see that numerical challenges are not restricted to Bayesian computation. We manually standardize the year covariate and see whether this helps—it does:

```

mny <- mean(data$year)
sdy <- sd(data$year)
cov1 <- (data$year - mny) / sdy
cov2 <- cov1 * cov1
cov3 <- cov1 * cov1 * cov1
glmm.fit <- lmer(C ~ (1 | yr) + cov1 + cov2 + cov3, family = poisson, data =
    data)
glmm.fit
Generalized linear mixed model fit by the Laplace approximation
Formula: C ~ (1 | yr) + cov1 + cov2 + cov3
Data: data
AIC      BIC logLik deviance
 63     71.44   -26.5        53

```

Random effects:

Groups	Name	Variance	Std.Dev.
yr	(Intercept)	0.0059868	0.077374

Number of obs: 40, groups: yr, 40

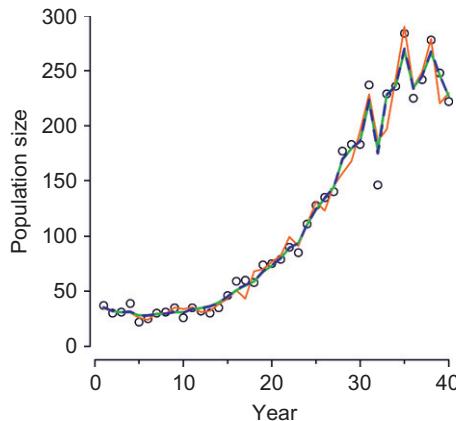
Fixed effects:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	4.33482	0.03412	127.05	< 2e-16 ***
cov1	1.22411	0.05627	21.75	< 2e-16 ***
cov2	0.06023	0.02686	2.24	0.0250 *
cov3	-0.23495	0.02934	-8.01	1.17e-15 ***
[ ... ]				

We form predictions and plot them (Fig. 4.2). If we wish, we can inspect the random-effects estimates by typing `ranef(glmm.fit)`. We could also have formed predictions by typing `fitted(glmm.fit)`, but doing this by hand enhances our understanding.

```
R.predictions <- exp(fixef(glmm.fit)[1] + fixef(glmm.fit)[2]*cov1 +
fixef(glmm.fit)[3]*cov2 + fixef(glmm.fit)[4]*cov3 +
unlist(ranef(glmm.fit)))
lines(data$year, R.predictions, col = "green", lwd = 2, type = "l")
```

Here is the WinBUGS solution. Remember that WinBUGS parameterizes the normal distribution in terms of the precision, that is, one over the variance.



**FIGURE 4.2** Simulated population size of peregrines in the French Jura over 40 years: expected population size (red), observed data (pair counts; black), estimated population trajectories from a frequentist (green, using REML in `lmer`), and a Bayesian analysis in WinBUGS (blue; posterior means) of a Poisson regression with cubic polynomial effects of year and year overdispersion. The R code to produce this figure slightly differs from the one shown in the book.

```

# Specify model in BUGS language
sink("GLMM_Poisson.txt")
cat("")
model {

# Priors
alpha ~ dunif(-20, 20)
beta1 ~ dunif(-10, 10)
beta2 ~ dunif(-10, 10)
beta3 ~ dunif(-10, 10)
tau <- 1 / (sd*sd)
sd ~ dunif(0, 5)

# Likelihood: note key components of a GLM in one line each
for (i in 1:n) {
  C[i] ~ dpois(lambda[i])           # 1. Distribution for random part
  log(lambda[i]) <- log.lambda[i]   # 2. Link function
  log.lambda[i] <- alpha + beta1 * year[i] + beta2 * pow(year[i],2) +
    beta3 * pow(year[i],3) + eps[i] # 3. Linear predictor incl.
                                         random year effect
  eps[i] ~ dnorm(0, tau)          # 4. Definition of random effects dist
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = data$C, n = length(data$C), year = cov1)

# Initial values
inits <- function() list(alpha = runif(1, -2, 2), beta1 = runif(1, -3, 3),
  sd = runif(1, 0,1))

# Parameters monitored
params <- c("alpha", "beta1", "beta2", "beta3", "lambda", "sd", "eps")

# MCMC settings
ni <- 30000
nt <- 10
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out<- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

```

Here are two important comments. First, when calling the `bugs()` function, we will from now on use so-called positional matching of the arguments. That is, unlike in Chapter 3, where we wrote `bugs(data = win.data, inits = inits, ...)`, we will now let the position of some arguments determine their identity, so the first argument is `data`, the second is the `inits` function, and so on. This achieves more succinct code. Second, we will no longer formally comment on convergence in every example. In reality, we always check for convergence by looking at the plots in WinBUGS and

by inspecting the values of Rhat in the summary produced by the `bugs()` function, we just do not say so.

```
# Summarize posteriors
print(out, dig = 2)
      mean    sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
alpha    4.33  0.04   4.26   4.31   4.33   4.36   4.40  1.00 1800
beta1   1.22  0.06   1.11   1.18   1.22   1.26   1.35  1.00  450
beta2   0.06  0.03   0.01   0.04   0.06   0.08   0.12  1.00 1300
beta3  -0.23  0.03  -0.30  -0.25  -0.23  -0.21  -0.18  1.01  360
lambda[1] 35.51  4.20  28.04  32.55  35.23  38.10  44.84  1.00 1700
lambda[2] 32.02  3.53  25.48  29.49  31.93  34.43  39.20  1.00 1200
lambda[3] 30.86  3.13  25.12  28.75  30.84  32.82  37.37  1.00 2600
[ ... ]
lambda[38] 267.75 14.22 241.29 257.80 267.30 277.40 296.30  1.00  660
lambda[39] 246.04 13.55 220.20 236.67 245.80 255.22 273.00  1.00 3000
lambda[40] 226.47 13.47 200.90 216.90 226.55 235.30 252.90  1.00 3000
sd       0.09  0.02   0.05   0.07   0.09   0.10   0.14  1.01  340
```

We get estimates that are fairly similar to those from the frequentist analysis using maximum likelihood. We plot the predicted population trajectory into Fig. 4.2.

```
WinBUGS.predictions <- out$mean$lambda
lines(data$year, WinBUGS.predictions, col = "blue", lwd = 2, type = "l",
      lty = 2)
```

Hence, overdispersion in counts can be accounted for by adding random effects at the data level (here, random year effects  $\epsilon_i$ ), assuming a normal distribution on the scale of the linear predictor for them, and estimating the spread of that normal distribution. Accounting for overdispersion more adequately accounts for the full uncertainty in the modeled system. We can see this by comparing the standard errors of the regression estimates under a simple GLM with those obtained under the GLMM. We quickly fit the respective models in R.

```
glm.fit <- glm(C ~ cov1 + cov2 + cov3, family = poisson, data = data)
summary(glm.fit)
summary(glmm.fit)
```

The uncertainty of all regression estimates is slightly increased under the GLMM compared with the GLM. Thus, the GLMM propagates the additional uncertainty in the modeled system into the regression estimates as it should. See also exercise 1 in Section 4.5.

### 4.2.2 Analysis of Real Data

Now, let us repeat these analyses for the real data from the French Jura. We need to standardize year to avoid worries with convergence.

```

# Read data again
peregrine <- read.table("falcons.txt", header = TRUE)

yr <- factor(peregrine$Year)
mny <- mean(peregrine$Year)
sdy <- sd(peregrine$Year)
cov1 <- (peregrine$Year - mny) / sdy
cov2 <- cov1 * cov1
cov3 <- cov1 * cov1 * cov1
glmm <- lmer(peregrine$Pairs ~ (1 | yr) + cov1 + cov2 + cov3, family =
    poisson, data = peregrine)
glmm
Generalized linear mixed model fit by the Laplace approximation
Formula: peregrine$Pairs ~ (1 | yr) + cov1 + cov2 + cov3
Data: peregrine
AIC  BIC logLik deviance
77.01 85.46 -33.51    67.01

Random effects:
Groups   Name        Variance Std.Dev.
yr       (Intercept) 0.0098814 0.099405
Number of obs: 40, groups: yr, 40

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 4.21205   0.03841 109.65  < 2e-16 ***
cov1        1.19085   0.06507  18.30  < 2e-16 ***
cov2        0.01720   0.02980   0.58   0.564
cov3       -0.27162   0.03361  -8.08  6.4e-16 ***
[ ... ]

```

We estimate an overdispersion standard deviation of about 0.1 (R also gives the square of that estimate, that is, the variance).

Next, we conduct the analysis in WinBUGS. We can reuse most ingredients of the analysis from [Section 4.2.1](#). Do not forget to check convergence; if there is trouble, you have to increase the chain length and/or the burnin period.

```

# Bundle data
win.data <- list(C = peregrine$Pairs, n = length(peregrine$Pairs),
    year = cov1)

# MCMC settings (may have to adapt)
ni <- 30000
nt <- 10
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
    n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
    bugs.dir, working.directory = getwd())

```

```
# Summarize posteriors
print(out, dig = 3)
      mean     sd   2.5%    25%    50%    75%  97.5% Rhat n.eff
alpha     4.210  0.043  4.120  4.181  4.211  4.239  4.291 1.002 1700
beta1    1.200  0.078  1.057  1.147  1.197  1.250  1.361 1.005 420
beta2    0.017  0.033 -0.048 -0.004  0.016  0.038  0.083 1.002 1500
beta3   -0.275  0.040 -0.356 -0.302 -0.274 -0.247 -0.200 1.005 440
lambda[1] 34.252 4.364 26.549 31.130 34.010 37.010 43.451 1.002 1400
lambda[2] 35.796 4.640 27.859 32.487 35.510 38.590 46.121 1.001 3000
lambda[3] 32.141 3.800 25.450 29.490 31.850 34.540 40.401 1.001 3000
[ ... ]
lambda[39] 180.358 12.146 157.600 171.775 180.050 188.600 205.100 1.001 3000
lambda[40] 176.702 12.751 152.597 168.100 176.300 185.300 202.402 1.001 3000
sd        0.117  0.032  0.059  0.095  0.116  0.136  0.182 1.002 3000
[ ... ]
```

As usual, we get similar estimates for both mean and variance parameters in R and WinBUGS. It is instructive to compare the posterior summaries from Poisson GLMM with those under the simple Poisson GLM in Section 3.2.2: you will see that the estimates of the mean parameters (`alpha`, `beta1`, `beta2`, and `beta3`) are not that different under both models. However, the uncertainty around them (posterior `sd`) is greater under the mixed Poisson model. Similarly, the 95% CRI are wider, and also the uncertainty around the `lambda` components is larger. Hence, our estimates properly account for the added uncertainty introduced by the second component of variation in the modeled system, the random year effects. Without accounting for the extra-Poisson variability in the counts, we would have underestimated parameter uncertainty.

### 4.3 MIXED MODELS WITH RANDOM EFFECTS FOR VARIABILITY AMONG GROUPS (SITE AND YEAR EFFECTS)

For the remainder of this chapter, we look at models for population counts that are indexed by both space and time, rather than only by time as in the preceding examples in this chapter. We will model counts  $C_{i,j}$  made at time  $i$  and site  $j$  as a function of cubic polynomials of year and start by assuming a Poisson distribution with expected count  $\lambda_{i,j}$ :

$$C_{i,j} \sim \text{Poisson}(\lambda_{i,j}).$$

To model spatiotemporal structure in the expected counts, we apply the usual log link function and express the log (expected count) as a function of covariates in a linear way:

$$\log(\lambda_{i,j}) = \alpha_j + \sum_k \beta_p * X_i^p + \epsilon_i$$

This equation describes a log-linear multiple regression of the expected count on an additive function of  $k=3$  time covariates  $X$ , that is, a fixed-effects Poisson GLM, with an added residual term  $\epsilon_i$ . Now, look at the indices of the parameters: the slope parameters (betas) do not vary by time or by among sites (they are not indexed by  $i$  or by  $j$ ): we estimate a single set of beta for all sites and times. However, the intercept  $\alpha$  is indexed by site ( $j$ ) and so codes for site effects, and the residual  $\epsilon$  is indexed by time ( $i$ ) and hence codes for time effects in addition to those of the covariate  $X$ .

The model, as written so far, is a fixed-effects model. The GLMMs considered in the rest of this chapter extend the fixed-effects Poisson GLM by adding two (or more) distributional assumptions to the equation above, for instance

$$\alpha_j \sim \text{Normal}(\mu, \sigma_\alpha^2) \text{ and}$$

$$\epsilon_i \sim \text{Normal}(0, \sigma_\epsilon^2).$$

This allows for stochastic site differences in the level of the population trajectory, that is, in the intercepts  $\alpha_j$ , and for stochastic year effects  $\epsilon_i$  common to all sites. Note that we could also estimate random year-by-site noise, say  $\epsilon_{i,j}$ , but we will assume here that our current model is sufficiently flexible. To describe each set of parameters,  $\alpha_j$  and  $\epsilon_i$ , we will estimate one (hyper)parameter that expresses the magnitude of the variability among sites ( $\sigma_\alpha^2$ ) and another for the variability among years ( $\sigma_\epsilon^2$ ).

Note that a simple reparameterization of this model would consist in “pulling out” the mean site effect,  $\mu$ , into an overall intercept. The site effects would then come from a mean-zero normal distribution rather than one with mean  $\mu$ .

$$\log(\lambda_{i,j}) = \mu + \sum_k \beta_p * X_i^p + \alpha_j + \epsilon_i$$

$$\alpha_j \sim \text{Normal}(0, \sigma_\alpha^2)$$

$$\epsilon_i \sim \text{Normal}(0, \sigma_\epsilon^2)$$

Models of this kind have become sort of standard for analyzing population counts (Link and Sauer, 2002; Ver Hoef and Jansen, 2007; Cressie et al., 2009, among many others). They are often called hierarchical models, since they contain a hierarchy of effects, and we can think of a nested relationship among the parameters in the model. However, the term, hierarchical model, by itself is about as informative about the actual model used as it would be to say that I am using a four-wheeled vehicle for locomotion; there are golf carts, quads, Smart cars, Volkswagen, and

Mercedes, for instance, and they all have four wheels. Similarly, plenty of statistical models applied in ecology have at least one set of random effects (beyond the residual) and therefore represent a hierarchical model. Hence, the term is not very informative about the particular model adopted for inference about an animal population.

### 4.3.1 Generation and Analysis of Simulated Data

For data simulation, we adapt the data generation function from Section 4.2.1 to several parallel time series of counts and to allow for random site and random year effects. We assume that there are site effects on the intercept of the trajectory only, not the slope parameters. So, we might think that this function generates replicate trajectories of counts of the peregrine population in the French Jura. We consider balanced data for convenience only; the models work also if there are missing values in the year-by-site data.

```
data.fn <- function(nsite = 5, nyear = 40, alpha = 4.18456, beta1 =
  1.90672, beta2 = 0.10852, beta3 = -1.17121, sd.site = 0.5, sd.year =
  0.2) {
  # nsite: Number of populations
  # nyear: Number of years
  # alpha, beta1, beta2, beta3: cubic polynomial coefficients of year
  # sd.site: standard deviation of the normal distribution assumed for
  #           the population intercepts alpha
  # sd.year: standard deviation of the normal distribution assumed for
  #           the year effects
  # We standardize the year covariate so that it runs from about -1 to 1
  # Generate data structure to hold counts and log(lambda)
  C <- log.expected.count <- array(NA, dim = c(nyear, nsite))

  # Generate covariate values
  year <- 1:nyear
  yr <- (year-20)/20 # Standardize
  site <- 1:nsite

  # Draw two sets of random effects from their respective distribution
  alpha.site <- rnorm(n = nsite, mean = alpha, sd = sd.site)
  eps.year <- rnorm(n = nyear, mean = 0, sd = sd.year)

  # Loop over populations
  for (j in 1:nsite) {

    # Signal (plus first level of noise): build up systematic part of
    # the GLM including random site and year effects
    log.expected.count[,j] <- alpha.site[j] + beta1 * yr + beta2 * yr^2
    + beta3 * yr^3 + eps.year
    expected.count <- exp(log.expected.count[,j])
  }
}
```

```

# Second level of noise: generate random part of the GLM: Poisson
# noise around expected counts
C[,j] <- rpois(n = nyear, lambda = expected.count)
}

# Plot simulated data
matplot(year, C, type = "l", lty = 1, lwd = 2, main = "", las = 1, ylab =
  "Population size", xlab = "Year")

return(list(nsites = nsites, nyears = nyears, alpha.site = alpha.site,
  beta1 = beta1, beta2 = beta2, beta3 = beta3, year = year, sd.site = sd.
  site, sd.year = sd.year, expected.count = expected.count, C = C))
}

```

We generate one very large data set and plot it (Fig. 4.3). Simulating 100 populations will result in a very large BUGS run time, and you may want to choose fewer sites to reduce the run time in your analysis, for example, `nsite = 10` will result in a BUGS run time (BRT) of about 12 min.

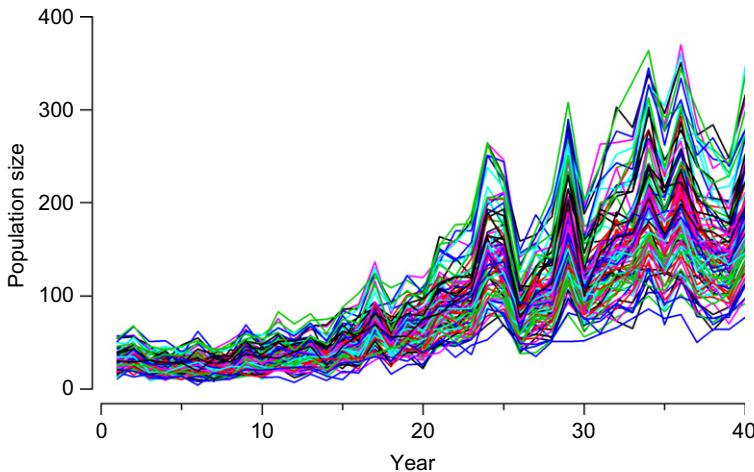
```
data <- data.fn(nsites = 100, nyears = 40, sd.site = 0.3, sd.year = 0.2)
```

We analyze using BUGS.

```

# Specify model in BUGS language
sink("GLMM_Poisson.txt")
cat("
model {

```



**FIGURE 4.3** One hundred replicate population trajectories with additive random site and year effects. We emphasize how a single stochastic process, here, the `data.fn()` function and in an ecologist's daily life, nature, can produce spectacularly different outcomes. In nature, we typically only ever see a single outcome of the stochastic process from which we want to infer the characteristics of the latter—a difficult challenge.

```

# Priors
for (j in 1:nsite) {
  alpha[j] ~ dnorm(mu, tau.alpha)           # 4. Random site effects
}
mu ~ dnorm(0, 0.01)                         # Hyperparameter 1
tau.alpha <- 1 / (sd.alpha*sd.alpha)        # Hyperparameter 2
sd.alpha ~ dunif(0, 2)
for (p in 1:3) {
  beta[p] ~ dnorm(0, 0.01)
}

tau.year <- 1 / (sd.year*sd.year)           # Hyperparameter 3
sd.year ~ dunif(0, 1)

# Likelihood
for (i in 1:nyear) {
  eps[i] ~ dnorm(0, tau.year)                # 4. Random year effects
  for (j in 1:nsite) {
    C[i,j] ~ dpois(lambda[i,j])              # 1. Distribution for random
                                                # part
    lambda[i,j] <- exp(log.lambda[i,j])       # 2. Link function
    log.lambda[i,j] <- alpha[j] + beta[1] * year[i] + beta[2] *
      pow(year[i],2) + beta[3] * pow(year[i],3) + eps[i] # 3. Linear
      predictor including random site and random year effects
  } #j
} #i
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = data$C, nsite = ncol(data$C), nyear = nrow(data$C),
                  year = (data$year-20) / 20) # Note year standardized

# Initial values
inits <- function() list(mu = runif(1, 0, 2), alpha = runif(data$nsite,
               -1, 1), beta = runif(3, -1, 1), sd.alpha = runif(1, 0, 0.1),
               sd.year = runif(1, 0, 0.1))

# Parameters monitored (may want to add "lambda")
params <- c("mu", "alpha", "beta", "sd.alpha", "sd.year")

# MCMC settings (may have to adapt)
ni <- 100000
nt <- 50
nb <- 50000
nc <- 3

# Call WinBUGS from R (BRT 98 min)
out <- bugs(win.data, inits, params, "GLMM_Poisson.txt", n.chains = nc,
            n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
            bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out, dig = 3)

```

Estimation in more complex mixed models can be challenging. Estimating variance parameters is always more difficult than estimating fixed-effects parameters. This is manifest in slower convergence rates of the Markov chains, longer run times or trouble in getting convergence at all. Problems are compounded by small sample sizes. For the estimation of a variance parameter, the relevant sample size is the number of groups among which we want to estimate the variability in some parameter. Estimating variance parameters with fewer than 5–10 groups can be particularly difficult.

### 4.3.2 Analysis of Real Data Set

We analyze data from the Swiss breeding bird survey MHB (“Monitoring Häufige Brutvögel”; Schmid et al., 2004), a scheme launched in 1999. During each breeding season, a systematic sample of about 300 1-km<sup>2</sup> quadrats is surveyed 2–3 times using the territory mapping method. This produces a count of putative bird territories at each site and year. We will use a sample of data from 235 sites over 9 years and model the annual territory counts of the coal tit (Fig. 4.4). We are interested in whether there is an overall population trend in Swiss coal tits and in the variation of counts among sites, years, and observers. In addition, we would like to see whether observers count fewer birds during their first year of survey in a particular quadrat.



**FIGURE 4.4** Coal tit (*Parus ater*), Finland, 2007. (Photograph by M. Varesvuo.)

Typically, each observer surveys only a small number of quadrats per year, but surveys the same quadrat(s) over a series of years.

We will fit a sequence of models starting with fixed effects only and then move on to including random effects. The comparison of analogous fixed- and random-effects models will be illuminating for your understanding of random effects. The most complex model entertained for count  $C_{i,j}$  in year  $i$  at site  $j$  will be as follows:

$C_{i,j} \sim \text{Poisson}(\lambda_{i,j})$	Data Distribution
$\log(\lambda_{i,j}) = \alpha_j + \beta_1 * \text{year}_i + \beta_2 * F_{i,j} + \delta_i + \gamma_{k(i,j)}$	Link Function and Linear Predictor
$\alpha_j \sim \text{Normal}(\mu, \sigma_\alpha^2)$	Random Site Effects
$\delta_i \sim \text{Normal}(0, \sigma_\delta^2)$	Random Year Effects
$\gamma_{k(i,j)} \sim \text{Normal}(0, \sigma_\gamma^2)$	Random Observer Effects

Thus, the log expected count,  $\log(\lambda_{i,j})$ , is a linear function of random site effects  $\alpha_j$ , random year effects  $\delta_i$ , and random observer effects  $\gamma_{k(i,j)}$ . In addition, there is a slope on year of magnitude  $\beta_1$  (i.e., a trend) and a change in the expected count of magnitude  $\beta_2$  if the indicator of first-year-of-service  $F_{i,j}$  is equal to 1. The three sets of random effects are assumed to be drawn from independent normal distributions whose variances (or standard deviations) we estimate along with a grand mean  $\mu$ .

```
# Read in the tit data and have a look at them
tits <- read.table("tits.txt", header = TRUE)
str(tits)
```

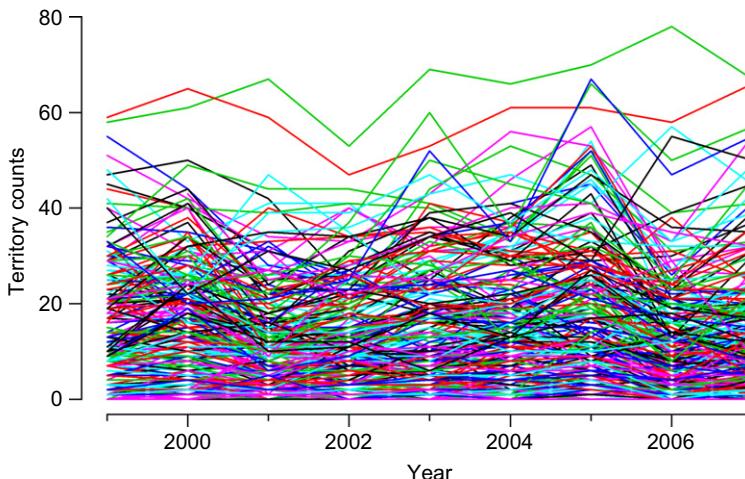
We have two environmental covariates (elevation and forest cover), annual territory counts 1999–2007 (y1999, etc.), an observer code (obs1999, etc., scaled to protect true observer ID), and finally a set of indicators for years when an observer surveyed a quadrat for the first time (first1999, etc.). Variable first1999 contains 1's only (except for some NAs), since MHB was launched in 1999. When modeling first-time observer effects, we may want to discard the 1999 data; otherwise the year 1999 effect is confounded with the first-time observer effect.

We collect counts, observer code, and first-time observer indicator variables in separate matrices and then plot the counts (Fig. 4.5; note use of the transposition function `t()`).

```
C <- as.matrix(tits[5:13])
obs <- as.matrix(tits[14:22])
first <- as.matrix(tits[23:31])

matplot(1999:2007, t(C), type = "l", lty = 1, lwd = 2, main = "",
       las = 1, ylab = "Territory counts", xlab = "Year", ylim = c(0, 80),
       frame = FALSE)
```

From Fig. 4.5, it appears that the Swiss coal tit metapopulation fluctuates around some fairly constant level. We will thus consider only a linear



**FIGURE 4.5** Territory counts of coal tits in 235-MHB quadrats in the Swiss breeding bird survey.

regression of counts on time rather than a higher order polynomial as for the French peregrines. However, we will try to explain variation among years and among sites by observer identity and first-year observer effects.

```
table(obs)
length(table(obs))
```

A total of 271 observers served from 1999 to 2007 and surveyed a fairly variable number of site years.

```
apply(first, 2, sum, na.rm = TRUE)
first1999 first2000 first2001 first2002 first2003 first2004 first2005
first2006 first2007
176      37      22      25      40      31      25      33      24
```

After 1999, about 10%–20% of observers are new on their quadrat each year. To model observer ID as a factor in WinBUGS, we need to recode it to be continuous.

```
a <- as.numeric(levels(factor(obs)))      # All the levels, numeric
newobs <- obs                            # Gets ObsID from 1:271
for (j in 1:length(a)) {newobs [which(obs==a[j])] <- j }
table(newobs)
```

We also need to fill up any missing values in the explanatory variates in WinBUGS (or else we have to specify priors for them). Unfortunately, there are two kinds of missing values in the observer ID data: one for site years when a quadrat was not surveyed (these correspond to NAs in the counts matrix C) and another when a quadrat was surveyed, but the observer

identity was not recorded. We impute some values in them, for example, 272 for missing observer identity and a zero in the first-year observer indicator variable. When transforming all NAs in the observer ID matrix to 272, we lump all unknown observers into a single observer identity. Given the large number of observers, this should hardly have an effect on the inference.

```
newobs[is.na(newobs)] <- 272
table(newobs)
first[is.na(first)] <- 0
table(first)
```

We are now ready to model these counts. Our strategy will be to start with the simplest possible model, with an intercept only, and then gradually build in more complexity. More specifically, we will first fit some fixed-effects models and only then we add the additional random-effects assumptions about some sets of previous fixed effects. This modification should clarify the exact meaning of the random-effects assumption. We will look at the deviance information criterion (DIC) as long as all effects in the model are fixed, but abandon it when we move to random effects, since the use of the standard DIC seems to be problematic then (Millar, 2009).

### **Null or Intercept-Only Model**

This model has a constant expected count throughout space and time.

```
# Specify model in BUGS language
sink("GLM0.txt")
cat("
model {

# Prior
alpha ~ dnorm(0, 0.01)      # log(mean count)

# Likelihood
for (i in 1:nyear) {
  for (j in 1:nsite) {
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- alpha
  } #j
} #i
}
", fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values
inits <- function() list(alpha = runif(1, -10, 10))
```

```

# Parameters monitored
params <- c("alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out0 <- bugs(win.data, inits, params, "GLM0.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out0, dig = 3)
      mean     sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
alpha    2.668 0.006   2.656   2.664   2.667   2.672   2.679     1 1500
deviance 32853.240 4.724 32850.000 32850.000 32850.000 32860.000 32860.000     1 1500
pD = 11.2 and DIC = 32864.4 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).

```

This cannot be a very good model, but it's a start. It says that the mean observed density of tits per 1 km<sup>2</sup> is  $\exp(2.67) = 14.4$ . Next, we add in fixed site effects.

### **Fixed Site Effects**

This model is essentially a one-way ANOVA, except that it is for a Poisson rather than a normal response.

```

# Specify model in BUGS language
sink("GLM1.txt")
cat("
model {

# Priors
for (j in 1:nsite){
  alpha[j] ~ dnorm(0, 0.01)      # Site effects
}

# Likelihood
for (i in 1:nyear){
  for (j in 1:nsite){
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- alpha[j]
  }  #j
}  #i
}
", fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

```

```

# Initial values (not required for all)
inits <- function() list(alpha = runif(235, -1, 1))

# Parameters monitored
params <- c("alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out1 <- bugs(win.data, inits, params, "GLM1.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out1, dig = 2)

      mean    sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
alpha[1] -0.17  0.36 -0.91 -0.40 -0.16  0.08  0.47 1.00 1500
alpha[2] -0.49  0.41 -1.36 -0.76 -0.45 -0.19  0.22 1.00 1200
alpha[3]  2.46  0.10  2.26  2.40  2.46  2.53  2.65 1.00  980
[ ... ]
alpha[233] 3.68  0.05  3.57  3.64  3.68  3.71  3.78 1.00 1500
alpha[234] 3.36  0.06  3.24  3.32  3.36  3.40  3.48 1.00 1500
alpha[235] 3.59  0.06  3.48  3.56  3.59  3.63  3.70 1.00  490
deviance 11688.71 21.82 11650.00 11670.00 11690.00 11700.00 11730.00 1.00 1500

pD = 238.3 and DIC = 11927 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).

```

Fitting site effects has greatly improved the fit of the model; the deviance has come down by half and so has the DIC. Next, we also add fixed year effects.

### **Fixed Site and Fixed Year Effects**

This model is a two-way, main-effects ANOVA for a Poisson response. Note how, in the inits function, we must not give an initial value for the first level of the year effects factor, which we have to set to zero to avoid overparameterization.

```

# Specify model in BUGS language
sink("GLM2.txt")
cat("
model {

# Priors
for (j in 1:nsite){          # Site effects
  alpha[j] ~ dnorm(0, 0.01)
}
for (i in 2:nyear){          # nyear-1 year effects
  eps[i] ~ dnorm(0, 0.01)
}
eps[1] <- 0                  # Aliased
}
```

```

# Likelihood
for (i in 1:nyear) {
  for (j in 1:nsite) {
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- alpha[j] + eps[i]
  } #j
} #i
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values
inits <- function() list(alpha = runif(235, -1, 1), eps = c(NA, runif(8,
-1, 1)))

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

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out2 <- bugs(win.data, inits, params, "GLM2.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out2, dig = 2)

      mean     sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
alpha[1]  -0.18  0.37   -0.96   -0.40   -0.16   0.07   0.49  1.00  1500
alpha[2]  -0.50  0.43   -1.37   -0.78   -0.47   -0.20   0.26  1.00  1500
alpha[3]   2.46  0.10    2.25    2.39    2.46    2.53   2.65  1.00  1000
[ ... ]
alpha[233]  3.67  0.06    3.56    3.64    3.67    3.71   3.78  1.00  920
alpha[234]  3.36  0.06    3.23    3.32    3.36    3.40   3.48  1.00  460
alpha[235]  3.59  0.06    3.47    3.55    3.59    3.63   3.69  1.00  1500
eps[2]     0.08  0.03    0.03    0.06    0.08    0.09   0.13  1.02   85
eps[3]    -0.16  0.03   -0.21   -0.18   -0.16   -0.14  -0.10  1.03   69
eps[4]    -0.11  0.03   -0.15   -0.12   -0.11   -0.09  -0.05  1.04   59
eps[5]     0.06  0.03    0.01    0.04    0.06    0.07   0.11  1.02   97
eps[6]     0.10  0.02    0.05    0.08    0.10    0.12   0.15  1.03   76
eps[7]     0.22  0.02    0.18    0.21    0.22    0.24   0.27  1.02  110
eps[8]    -0.16  0.03   -0.21   -0.18   -0.16   -0.14  -0.11  1.02   98
eps[9]    -0.07  0.03   -0.12   -0.08   -0.07   -0.05  -0.02  1.02   94
deviance 11237.79 22.12 11200.00 11220.00 11240.00 11250.00 11280.00 1.00 1500

pD = 244.7 and DIC = 11482.4 (using the rule, pD = var(deviance)/2)
DIC is an estimate of expected predictive error (lower deviance is better).

```

With the introduction of year effects, the deviance and the DIC have gone down quite a bit again, so there seem to be annual population fluctuations. Now, we increase the model complexity by specifying random instead of fixed effects. Watch how simple the move from fixed to random effects is when a model is defined in the BUGS language. Essentially, all that is required is a common distribution to be assumed for a set of grouped effects with hyperparameters that are going to be estimated and that therefore need hyperpriors in turn. We start with the random site model.

### **Random Site Effects (No Year Effects)**

The linear predictor of this model is that of a one-way, random-effects ANOVA.

```
# Specify model in BUGS language
sink("GLMM1.txt")
cat("
model {

# Priors
for (j in 1:nSITE) {
    alpha[j] ~ dnorm(mu.alpha, tau.alpha)      # Random site effects
}
mu.alpha ~ dnorm(0, 0.01)
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

# Likelihood
for (i in 1:NYEAR) {
    for (j in 1:nSITE) {
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- alpha[j]
    } #j
} #i
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyyear = ncol(C))

# Initial values
inits <- function() list(mu.alpha = runif(1, 2, 3))

# Parameters monitored
params <- c("alpha", "mu.alpha", "sd.alpha")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3
```

```
# Call WinBUGS from R (BRT < 1 min)
out3 <- bugs(win.data, inits, params, "GLMM1.txt", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out3, dig = 2)

      mean     sd    2.5%   25%   50%   75%  97.5% Rhat n.eff
alpha[1] -0.03  0.33  -0.69  -0.25 -0.02  0.21  0.57 1.00 1500
alpha[2] -0.26  0.36  -1.03  -0.49 -0.24  0.00  0.40 1.00 1500
alpha[3]  2.46  0.11   2.25   2.39  2.46  2.52  2.67 1.00 1500
[ ... ]
alpha[233] 3.68  0.05   3.57   3.64  3.67  3.71  3.78 1.00 1500
alpha[234] 3.36  0.06   3.24   3.32  3.36  3.40  3.48 1.00 1500
alpha[235] 3.59  0.06   3.48   3.55  3.59  3.63  3.69 1.00  650
mu.alpha  2.09  0.09   1.93   2.03  2.09  2.15  2.27 1.00 1500
sd.alpha   1.33  0.06   1.21   1.28  1.32  1.37  1.46 1.00 1500
deviance 11692.91 22.38 11650.00 11680.00 11690.00 11710.00 11740.00 1.00  650
```

The mass of the posterior distribution of the standard deviation of the random site effects is concentrated well away from zero. This confirms our previous conclusion that sites differ substantially in their expected counts of coal tits. We next add a set of random year effects and in addition reparameterize the model to have a single grand mean. Each set of random effects is then centered around that grand mean, which helps convergence.

### **Random Site and Random Year Effects**

This linear model corresponds to a main-effects ANOVA with two random factors. In comparison to the fixed-effects counterpart of this model (GLM 2), we no longer need to constrain one effect of one factor to zero to avoid overparameterization. The borrowing strength among parameters within the same random-effects factor ensures that all can be estimated.

```
# Specify model in BUGS language
sink("GLMM2.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                                # Grand mean
for (j in 1:nsite){
  alpha[j] ~ dnorm(0, tau.alpha)                      # Random site effects
}
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

for (i in 1:nyear){
  eps[i] ~ dnorm(0, tau.eps)                          # Random year effects
}
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 3)
```

```

# Likelihood
for (i in 1:nyear) {
  for (j in 1:nsite) {
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- mu + alpha[j] + eps[i]
  } #j
} #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C))

# Initial values (not required for all)
inits <- function() list(mu = runif(1, 0, 4), alpha = runif(235, -2, 2),
  eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "alpha", "eps", "sd.alpha", "sd.eps")

# MCMC settings
ni <- 6000
nt <- 5
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
out4 <- bugs(win.data, inits, params, "GLMM2.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out4, dig = 2)

      mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
mu     2.09  0.10   1.89   2.03   2.10   2.16   2.29 1.01   860
alpha[1] -2.13  0.34  -2.83  -2.35  -2.12  -1.90  -1.51 1.00  3000
alpha[2] -2.36  0.37  -3.14  -2.60  -2.35  -2.11  -1.68 1.00  2100
alpha[3]  0.36  0.13   0.10   0.27   0.36   0.45   0.62 1.00   590
[...]
alpha[233]  1.57  0.10   1.38   1.51   1.57   1.64   1.76 1.01   260
alpha[234]  1.26  0.10   1.05   1.19   1.26   1.33   1.46 1.00   770
alpha[235]  1.49  0.10   1.30   1.42   1.49   1.55   1.68 1.01   310
eps[1]     0.00  0.06  -0.12  -0.03   0.00   0.04   0.12 1.03   88
eps[2]     0.08  0.06  -0.04   0.04   0.08   0.11   0.19 1.03   86
eps[3]    -0.16  0.06  -0.27  -0.19  -0.15  -0.12  -0.04 1.03   93
eps[4]    -0.10  0.06  -0.22  -0.14  -0.10  -0.07  0.01 1.04   67
eps[5]     0.06  0.06  -0.06   0.03   0.06   0.10   0.17 1.03   89
eps[6]     0.10  0.06  -0.02   0.06   0.10   0.13   0.21 1.03   89
eps[7]     0.22  0.06   0.11   0.19   0.22   0.26   0.34 1.05   77
eps[8]    -0.16  0.06  -0.28  -0.19  -0.16  -0.12  -0.05 1.03   83
eps[9]    -0.06  0.06  -0.18  -0.10  -0.06  -0.03   0.05 1.03   84
sd.alpha   1.33  0.07   1.21   1.28   1.33   1.37   1.47 1.00  3000
sd.eps    0.15  0.05   0.09   0.12   0.14   0.18   0.28 1.00  1100
deviance 11240.99 22.57 11200.00 11230.00 11240.00 11260.00 11290.00 1.00  3000

```

The year effects do not seem to be very large; at any rate, they (`sd.eps`) are much smaller than the variation of counts among sites (`sd.alpha`). Next, we add a fixed first-year observer effect.

### **Random Site and Random Year Effects and First-Year Fixed Observer Effect**

This model is analogous to a three-way ANOVA with two factors random and one fixed and all of them acting in an additive way.

```
# Specify model in BUGS language
sink("GLMM3.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                                # Overall mean
beta2 ~ dnorm(0, 0.01)                               # First-year observer effect

for (j in 1:nsite){
    alpha[j] ~ dnorm(0, tau.alpha)                 # Random site effects
}
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)
for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.eps)                     # Random year effects
}
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 5)

# Likelihood
for (i in 1:nyear){
    for (j in 1:nsite){
        C[i,j] ~ dpois(lambda[i,j])
        lambda[i,j] <- exp(log.lambda[i,j])
        log.lambda[i,j] <- mu + beta2 * first[i,j] + alpha[j] + eps[i]
    } #j
} #i
}, fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), first = t
(first))

# Initial values
inits <- function() list(mu = runif(1, 0, 4), beta2 = runif(1, -1, 1),
alpha = runif(235, -2, 2), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "beta2", "alpha", "eps", "sd.alpha", "sd.eps")

# MCMC settings
ni <- 6000
```



```

for (j in 1:nsite){
  alpha[j] ~ dnorm(0, tau.alpha)    # Random site effects
}
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 5)

for (i in 1:nyear){
  eps[i] ~ dnorm(0, tau.eps)        # Random year effects
}
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 3)

# Likelihood
for (i in 1:nyear){
  for (j in 1:nsite){
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- mu + betal * year[i] + beta2 * first[i,j] +
      alpha[j] + eps[i]
    } #j
  } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), first = t
  (first), year = ((1:9)-5) / 4)

# Initial values
inits <- function() list(mu = runif(1, 0, 4), betal = runif(1, -1, 1),
  beta2 = runif(1, -1, 1), alpha = runif(235, -2, 2), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "betal", "beta2", "alpha", "eps", "sd.alpha",
  "sd.eps")

# MCMC settings
ni <- 12000
nt <- 6
nb <- 6000
nc <- 3

# Call WinBUGS from R (BRT 7 min)
out6 <- bugs(win.data, inits, params, "GLMM4.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out6, dig = 2)
      mean     sd    2.5%     25%     50%     75%   97.5% Rhat n.eff
mu       2.09  0.10    1.88    2.02    2.10    2.17    2.28 1.06   64
betal    0.00  0.08   -0.15   -0.06   -0.01    0.04    0.18 1.03  210
beta2    0.00  0.02   -0.04   -0.02    0.00    0.01    0.03 1.00 1500
alpha[1]  -2.12  0.34   -2.82   -2.34   -2.10   -1.88   -1.52 1.00  610
alpha[2]  -2.36  0.37   -3.15   -2.60   -2.34   -2.11   -1.67 1.00 3000
alpha[3]   0.36  0.13    0.10    0.27    0.36    0.45    0.63 1.02  110
[ ... ]

```

alpha[233]	1.58	0.10	1.39	1.51	1.57	1.65	1.78	1.04	67
alpha[234]	1.26	0.11	1.06	1.19	1.26	1.34	1.48	1.03	81
alpha[235]	1.49	0.10	1.30	1.42	1.49	1.57	1.70	1.03	74
eps[1]	0.00	0.10	-0.19	-0.06	0.00	0.07	0.20	1.02	140
eps[2]	0.07	0.08	-0.09	0.02	0.07	0.13	0.24	1.02	150
eps[3]	-0.16	0.07	-0.30	-0.20	-0.16	-0.11	-0.01	1.02	150
eps[4]	-0.10	0.06	-0.23	-0.14	-0.10	-0.07	0.02	1.02	170
eps[5]	0.06	0.06	-0.05	0.02	0.06	0.10	0.17	1.02	170
eps[6]	0.10	0.06	-0.03	0.07	0.10	0.14	0.21	1.01	260
eps[7]	0.22	0.07	0.07	0.18	0.23	0.27	0.35	1.01	270
eps[8]	-0.15	0.08	-0.34	-0.20	-0.15	-0.10	-0.01	1.01	270
eps[9]	-0.06	0.10	-0.29	-0.12	-0.05	0.01	0.12	1.02	290
sd.alpha	1.33	0.07	1.21	1.28	1.33	1.37	1.46	1.00	3000
sd.eps	0.17	0.06	0.09	0.13	0.16	0.19	0.30	1.01	650
deviance	11241.89	22.31	11200.00	11230.00	11240.00	11260.00	11290.00	1.00	3000

### The Full Model

Finally, we fit the full model with random site effects, an overall linear time trend, random observer effects, random year effects, and a fixed first-year observer effect. Based on the estimates in the previous model, we constrain the random effects standard deviations sufficiently (i.e., specify a smaller range for the uniform prior distributions) so that WinBUGS does not easily get lost numerically. We also increase chain length.

```
# Specify model in BUGS language
sink("GLMM5.txt")
cat("
model {

# Priors
mu ~ dnorm(0, 0.01)                                # Overall intercept
beta1 ~ dnorm(0, 0.01)                               # Overall trend
beta2 ~ dnorm(0, 0.01)                               # First-year observer effect

for (j in 1:nsite){
    alpha[j] ~ dnorm(0, tau.alpha)      # Random site effects
}
tau.alpha <- 1/ (sd.alpha * sd.alpha)
sd.alpha ~ dunif(0, 3)

for (i in 1:nyear){
    eps[i] ~ dnorm(0, tau.eps)        # Random year effects
}
tau.eps <- 1/ (sd.eps * sd.eps)
sd.eps ~ dunif(0, 1)

for (k in 1:nobs){
    gamma[k] ~ dnorm(0, tau.gamma)    # Random observer effects
}
tau.gamma <- 1/ (sd.gamma * sd.gamma)
sd.gamma ~ dunif(0, 1)
```

```

# Likelihood
for (i in 1:nyear) {
  for (j in 1:nsite) {
    C[i,j] ~ dpois(lambda[i,j])
    lambda[i,j] <- exp(log.lambda[i,j])
    log.lambda[i,j] <- mu + betal * year[i] + beta2 * first[i,j] +
      alpha[j] + gamma[newobs[i,j]] + eps[i]
  } #j
} #i
}, fill = TRUE)
sink()

# Bundle data
win.data <- list(C = t(C), nsite = nrow(C), nyear = ncol(C), nobs = 272,
  newobs = t(newobs), first = t(first), year = ((1:9)-5) / 4)

# Initial values
inits <- function() list(mu = runif(1, 0, 4), betal = runif(1, -1, 1),
  beta2 = runif(1, -1, 1), alpha = runif(235, -1, 1), gamma = runif(272,
  -1, 1), eps = runif(9, -1, 1))

# Parameters monitored
params <- c("mu", "betal", "beta2", "alpha", "gamma", "eps",
  "sd.alpha", "sd.gamma", "sd.eps")

# MCMC settings
ni <- 12000
nt <- 6
nb <- 6000
nc <- 3

# Call WinBUGS from R (BRT 11 min)
out7 <- bugs(win.data, inits, params, "GLMM5.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out7, dig = 2)
      mean     sd    2.5%   25%   50%   75%  97.5% Rhat n.eff
mu      2.08  0.10    1.87   2.01   2.07   2.14   2.27 1.02  570
betal    0.02  0.09   -0.16  -0.04   0.01   0.07   0.24 1.04 3000
beta2    0.02  0.02   -0.02   0.01   0.02   0.04   0.07 1.00 3000
alpha[1] -2.43  0.37  -3.18  -2.66  -2.41  -2.18  -1.75 1.00  790
alpha[2] -2.10  0.41  -2.92  -2.37  -2.08  -1.81  -1.33 1.01  210
alpha[3]  0.36  0.26  -0.15   0.20   0.36   0.53   0.87 1.01  400
[...]
alpha[233] 1.63  0.30    1.04   1.43   1.62   1.83   2.25 1.01  300
alpha[234] 1.55  0.20    1.17   1.41   1.54   1.69   1.94 1.04  60
alpha[235] 1.78  0.20    1.41   1.64   1.77   1.92   2.17 1.04  61
gamma[1]   0.06  0.25   -0.44  -0.10   0.06   0.23   0.58 1.00 3000
gamma[2]  -0.01  0.34   -0.70  -0.24  -0.02   0.22   0.65 1.00  610
gamma[3]  -0.50  0.27   -1.05  -0.68  -0.50  -0.32   0.00 1.00 1800
[...]

```

gamma [270]	-0.57	0.28	-1.15	-0.75	-0.56	-0.38	-0.03	1.00	1600
gamma [271]	0.02	0.23	-0.44	-0.14	0.02	0.17	0.48	1.00	1300
gamma [272]	-0.27	0.20	-0.66	-0.41	-0.28	-0.13	0.08	1.07	36
eps [1]	0.01	0.10	-0.19	-0.05	0.01	0.07	0.24	1.03	220
eps [2]	0.10	0.09	-0.08	0.04	0.09	0.15	0.28	1.03	130
eps [3]	-0.14	0.07	-0.29	-0.18	-0.14	-0.10	0.01	1.03	75
eps [4]	-0.09	0.06	-0.22	-0.13	-0.10	-0.06	0.02	1.04	55
eps [5]	0.06	0.06	-0.06	0.02	0.06	0.09	0.16	1.05	45
eps [6]	0.09	0.06	-0.04	0.06	0.09	0.13	0.21	1.06	52
eps [7]	0.21	0.08	0.05	0.16	0.21	0.25	0.35	1.06	66
eps [8]	-0.16	0.09	-0.36	-0.22	-0.15	-0.10	0.02	1.06	110
eps [9]	-0.07	0.11	-0.30	-0.14	-0.06	0.00	0.14	1.06	130
sd.alpha	1.31	0.07	1.18	1.26	1.30	1.35	1.45	1.00	3000
sd.gamma	0.34	0.03	0.28	0.32	0.34	0.36	0.40	1.00	640
sd.eps	0.17	0.06	0.09	0.13	0.15	0.19	0.32	1.02	130
deviance	10687.47	30.28	10630.00	10670.00	10690.00	10710.00	10750.00	1.00	2200

It appears that there is much variation in the counts of coal tits due to differences among sites and also due to differences among observers. We note, though, that site and observer effects are confounded to some degree, since only one to a few observers are tested on each site and that observers are not randomly allocated to sites. There is a fairly small variance component due to years. Neither year as a continuous covariate (beta1) nor first-year observer effects (beta2) seem important, since their 95% CRI covers zero by a wide margin. Regarding the latter, we repeat that first-year observer effects are confounded with the effect of the first year. For a more thorough assessment of first-year observer effects, we might want to repeat the analysis for years 2–9 only.

This concludes our overview of different variants of a Poisson GLMM. We hasten to say that the models in this chapter are purely phenomenological and not necessarily the best models for animal counts. For instance, we have not built in any population dynamics: changes from year to year are not autocorrelated as one might expect owing to the effects of density dependence or of the vital rates inducing these changes. For models that include such additional ecological realism, see Chapters 5 and 11.

## 4.4 SUMMARY AND OUTLOOK

In this chapter, we have introduced another crucial concept (besides the GLM) of applied statistical models: random effects. We have seen that the inclusion of random effects changes the scope of inference in an analysis, allows one to study the variability in parameter sets, accounts for internal structure (correlations) in the data, achieves a more honest accounting for the uncertainties in a modeled system, and may result in improved estimates of each parameter owing to “borrowing strength

from the ensemble". We hope that the simulation of data sets and model specification in the BUGS language has enhanced your understanding of the actual meaning of random effects, something which we believe is easily lost when specifying random-effects models in software such as R or GenStat. The setting of all random effects in this chapter was the generalized linear model (GLM); hence, we dealt with GLMMs. Poisson GLMMs now form one of the standard models for counts of animals; see, for example, the papers by Link and Sauer (2002), Sauer and Link (2002), Ver Hoef and Jansen (2007), and Cressie et al. (2009). They are hierarchical models and to some (e.g., Cressie et al., 2009) have become almost synonymous with "the hierarchical model" in ecology. However, there are many different hierarchical models in ecology, and we will encounter a variety of hierarchical or random-effects models in almost every chapter in this book.

The Poisson GLMMs in this chapter are useful, for instance, because they more properly partition the effects of multiple sources of variability in a modeled system. One way how a partitioning of the system variability can be achieved is to separate out variability intrinsic to the studied system and variability associated with the measurement, or the observation, of that system. Hence, Poisson GLMMs achieve a certain degree of partitioning of the data into what may be called the ecological process and the observation process.

However, we believe that these models have important shortcomings as a general framework for inference about animal or plant population sizes. Most of all, they cannot directly account for that hallmark of ecological field data, namely, imperfect detection. Models such as those in this chapter do not contain an explicit description of a meaningful ecological process; they are "implicit" hierarchical models in the sense of Royle and Dorazio (2008). Their state is some sort of "expected count", and it is difficult to attach a clear biological interpretation to that. In other words, the expected count is always a result of a certain observation process and will depend to a large extent on how good a job you do at the counting. This can hardly be claimed to be of ecological interest!

A variant of the implicit hierarchical models in this chapter has come to be called "state-space models" in population dynamics and is the subject of the next chapter. They go one step further in introducing biological realism into the modeling, in that they make a better distinction between what they call process and observation models. Furthermore, their process model usually contains added biological realism in the form of an auto-regressive population model and may contain a parameter for density dependence. These are powerful and interesting models, and yet, they fail to explicitly account for imperfect detection and its effects on the inference about the population processes. To make a clear and ecologically meaningful distinction between the ecological and the observation

processes, other protocols and models must be chosen, leading to what Royle and Dorazio (2008) called “explicit” hierarchical models. These models exploit explicit information about the observation process, enabling one to explicitly model the observation process by estimating detection probability. In the context of statistical inference about animal numbers, the natural generalization of an implicit Poisson GLMM is the class of binomial mixture models introduced in Chapter 12.

## 4.5 EXERCISES

---

1. Overdispersion: Generate a data set using the function in Section 4.2.1 and use WinBUGS to compare the regression estimates under the Poisson GLM and those under the Poisson GLMM. The Bayesian analysis yields better estimates of the uncertainty in the estimates of a random-effects model and lets you see more clearly how the regression estimates have an increased posterior standard deviation when estimated under the model with random year effects.
2. First-year observer effect: We have seen that in the tit data, any first-year observer effect is confounded with the effect of the first year. Repeat the last analysis for a restricted data set without year 1.
3. Reparameterizations: In GLMM 2, put the grand mean of the double random-effects model,  $\mu$ , into the hyperdistribution of one of the random effects, that is, fit the model like the following:

```
for (j in 1:nsite) {
  alpha[j] ~ dnorm(mu, tau.alpha)
}
```

You will see that convergence is worse. This is an example of where WinBUGS is very sensitive to how a model is parameterized.

4. Interpretation of random effects: Fit a series of models to the tit data with different random effects:
  - A site random effect: random contributions from each site
  - A year random effect: random contributions from each year
  - A site plus a year random effect
  - A site-by-year random effect: random contributions from each site–year combination

Compare parameter estimates, explain the difference in the interpretation of those models, and try to make sense of the differences.

5. Fixed and random: Convert these models into fixed-effects models, that is, specify each of the following models without making the assumption that a set of effects come from a common distribution: site, year, site + year, observer, site + observer, year + observer,

site + year + observer, first-year indicator. Comparing the fixed- and the random-effects version of a model as specified in the BUGS language will be very helpful for your understanding of mixed models!

6. Covariates: In the tit model in [Section 4.3.2](#), add the log-linear effects of elevation and forest cover in the linear predictor of abundance. Also add in squared effects of these covariates.
7. Take the model and the data from [Section 4.3.1](#).
  - a. Drop the quadratic and the cubic polynomial terms of year.
  - b. Next, turn the random-effects Poisson GLM into a fixed- (year and site) effects Poisson GLM, that is, drop the randomness assumption for site and year. Hint: your model will then be a two-way, main-effects ANOVA, so you will have to constrain some parameters to make it identifiable.
8. Take the model and the data from [Section 4.3.1](#) and model the response as coming from a normal distribution. This will clarify some of the differences between a normal and a Poisson GLMM.

# State-Space Models for Population Counts

## OUTLINE

5.1 Introduction	115
5.2 A Simple Model	118
5.3 Systematic Bias in the Observation Process	121
5.4 Real Example: House Martin Population Counts in the Village of Magden	126
5.5 Summary and Outlook	131
5.6 Exercises	131

## 5.1 INTRODUCTION

State-space models are hierarchical models that decompose an observed time series of counts or other observed responses into a process variation and an observation error component. They are suitable for the description of Markovian, that is, autoregressive, processes that are latent or hidden, because they are observed imperfectly (Harvey, 1989). Originally developed in industry (Kalman, 1960), the use of state-space models in population dynamics has only recently begun (De Valpine and Hastings, 2002; Buckland et al., 2004; Clark and Björnstad, 2004). The change in

population size over time is a Markovian process because population size in year  $t + 1$  depends on population size in year  $t$ . Perhaps the simplest population dynamics model is the exponential:

$$N_{t+1} = N_t \lambda_t,$$

where  $\lambda_t$  is the population growth rate, defined for  $t = 1, \dots, T - 1$ ;  $T$  being the number of years with observed counts. A typical goal of a population dynamics analysis is to estimate the population growth rate and to study factors affecting it. This is relatively straightforward if the exact population size in each year is known. Regression-like models can then be used to estimate temporal variability, the strength of density-dependence, and the impact of environmental factors (see, e.g., Dennis et al., 1991; Dennis and Taper, 1994; Lande et al., 2003). However, only in exceptional cases do we know the exact size of a population. More typically, we only have a count with some unknown observation error associated. This observation error must be accounted for when analyzing population dynamics data. Otherwise, the power to detect factors affecting population dynamics is reduced and density-dependence may be detected spuriously (Freckleton et al., 2006).

The state-space models of this chapter allow one to deal with the challenge of observation error in the analysis of population dynamics, at least partially (see [Section 5.3](#)). The state-space model adopted here consists of two sets of equations. The state-process equations describe the true, but unknown development of a state, that is, the population dynamics free of observation error. For the dynamics of an exponentially growing population, the unknown state is population size  $N_t$  at time  $t$ :

$$N_{t+1} = N_t \lambda_t$$

$$\lambda_t \sim \text{Normal}(\bar{\lambda}, \sigma_\lambda^2).$$

In this model, we assume that the growth rates ( $\lambda_t$ ) are realizations of a normal random process with mean  $\bar{\lambda}$  and variance  $\sigma_\lambda^2$ . The mean is the long-term growth rate of the population, and the variance is a measure of the environmental variability affecting this growth rate, that is, environmental stochasticity. Note that the population size in the first year is not defined by the equations above. We therefore need to define this initial state by another model. In the Bayesian context, we can either specify a prior distribution for  $N_1$ , or fix it to the observed count, making the assumption that the first count is error-free.

The second set of equations maps the true state of the process on the observed data. These observation equations are conditional on the process equations. Assume we have a population count for each of a series of years.

We link these observations ( $y_t$ ) with the true population size using the following equations:

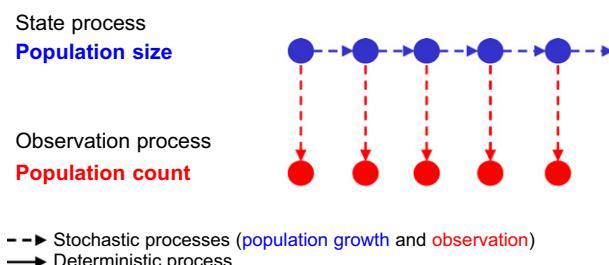
$$y_t = N_t + \varepsilon_t$$

$$\varepsilon_t \sim \text{Normal}(0, \sigma_y^2).$$

They represent the assumption that in year  $t$ , we make an observation error of magnitude  $\varepsilon_t$  and that these errors in the counts around the true population size can be described by a normal distribution with mean zero and variance  $\sigma_y^2$ ; the latter is the observation error. In this model, the true population size is as likely to be over- as it is to be undercounted. For instance, in one year, we may miss more individuals than we double-count, resulting in a negative value of  $\varepsilon_t$ , and in another year, it may be the other way round. The observation error is, in fact, a residual error and therefore incorporates not only observation errors but also the lack of fit of the state equations. Figure 5.1 represents the model graphically.

The model assumes that the two sets of random terms,  $\lambda_t$  and  $\varepsilon_t$ , are serially independent, independent from each other and that each set is identically distributed. Formulations of the likelihood of the model can be found, for example, in De Valpine and Hastings (2002). The model likelihood is composed of the likelihoods for the initial state, for the state process, and for the observation process.

The state-space modeling framework is very flexible and can be extended in various ways. We could, of course, define different state-processes (e.g., include survival and fecundity, see Chapter 11, or use a logistic growth model). Also, different descriptions of the observation process are possible, for instance, the assumption of binomial, Poisson, or lognormal errors. As we will see later, state-space models can be used for purposes other than estimating the true trajectories of population size. Examples we will see later include probabilities of survival (Chapters 7, 8, and 10), state-transition (Chapter 9),



**FIGURE 5.1** State process of a population and the conditional observation process. The state process describes the dynamics of population size over time ( $N_t$ ), whereas the observation process describes the error-prone population counts ( $y_t$ ). Note that this figure is essentially a version of Fig. 2.1.

and species occurrence and site occupancy (Chapter 12). They are also at the heart of integrated population models (Chapter 11). Because the observation process is conditional on the state process, a state-space model is also a hierarchical model (Royle and Dorazio, 2008).

## 5.2 A SIMPLE MODEL

We will first simulate and analyze count data in a very simple context. Let's assume that a population of ibex (Fig. 5.2) with initial size of 30 individuals was monitored during 25 years and grew annually by 2% on average. Shortly after the start of snow-melt, ibexes are counted by the aid of a telescope from the slope opposite to the mountain ridge which constitutes the core area of the population. The population survey is not perfect, and we assume that the variance of the observation error is 20. To simulate the data, we first define the underlying parameters.

```
n.years <- 25          # Number of years
N1 <- 30                # Initial population size
mean.lambda <- 1.02      # Mean annual population growth rate
sigma2.lambda <- 0.02    # Process (temporal) variation of the growth rate
sigma2.y <- 20           # Variance of the observation error
```

Next we simulate the population sizes under the assumption of exponential growth.



**FIGURE 5.2** Ibexes (*Capra ibex*), Switzerland (Photograph by F. Labhardt).

```

y <- N <- numeric(n.years)
N[1] <- N1
lambda <- rnorm(n.years-1, mean.lambda, sqrt(sigma2.lambda))
for (t in 1:(n.years-1)) {
  N[t+1] <- N[t] * lambda[t]
}

```

Finally, we generate the observed data conditional on the true population sizes.

```

for (t in 1:n.years) {
  y[t] <- rnorm(1, N[t], sqrt(sigma2.y))
}

```

You may have noticed that neither population size nor the counts are integers, as might be expected for a biological population. However, our goal here is to simulate and analyze the data in exactly the same way as the classical state-space models of this chapter are formulated and to explore the performance of state-space models. In Section 5.3, we will simulate a data set under more mechanistic assumptions which lead to integers.

Now, we analyze the simulated data. We first write the model in the BUGS language.

```

# Specify model in BUGS language
sink("ssm.bug")
cat("
model {

# Priors and constraints
N.est[1] ~ dunif(0, 500)          # Prior for initial population size
mean.lambda ~ dunif(0, 10)         # Prior for mean growth rate
sigma.proc ~ dunif(0, 10)          # Prior for sd of state process

sigma2.proc <- pow(sigma.proc, 2)
tau.proc <- pow(sigma.proc, -2)
sigma.obs ~ dunif(0, 100)          # Prior for sd of observation process
sigma2.obs <- pow(sigma.obs, 2)
tau.obs <- pow(sigma.obs, -2)

# Likelihood
# State process
for (t in 1:(T-1)) {
  lambda[t] ~ dnorm(mean.lambda, tau.proc)
  N.est[t+1] <- N.est[t] * lambda[t]
}

# Observation process
for (t in 1:T) {
  y[t] ~ dnorm(N.est[t], tau.obs)
}
", fill = TRUE)
sink()

```

We have chosen uniform priors for the standard deviations of the process and observation errors. This is to be preferred over a gamma prior distribution of the variances because they are less informative (Gelman, 2006). Next, we bundle the data, define initial values and MCMC settings, and analyze the model. The initial value for the population size in the first year needs to be chosen with care; if it is too far from the true value, the model may not run. A practical option is to use a random value close to the population counts. Problems with updating in the MCMC algorithm may also occur if the initial values for the process and observation standard deviations are too large. Again, choosing “good” initial values helps to avoid problems.

```
# Bundle data
bugs.data <- list(y=y, T=n.years)

# Initial values
inits <- function(){list(sigma.proc = runif(1, 0, 5), mean.lambda =
  runif(1, 0.1, 2), sigma.obs = runif(1, 0, 10), N.est = c(runif(1, 20,
  40), rep(NA, (n.years-1))))}

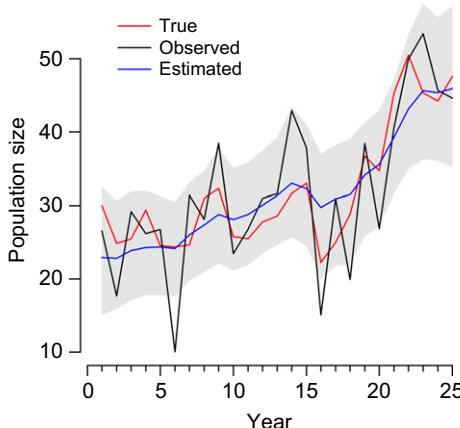
# Parameters monitored
parameters <- c("lambda", "mean.lambda", "sigma2.obs", "sigma2.proc",
  "N.est")

# MCMC settings
ni <- 25000
nt <- 3
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
ssm <- bugs(bugs.data, inits, parameters, "ssm.bug", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
  bugs.dir, working.directory=getwd())
```

How quickly the Markov chains converge depends greatly on the data. With some data sets, convergence is obtained much more swiftly than with others. Now we draw a plot (Fig. 5.3) to compare estimated, true, and observed population sizes. We notice that the estimated population sizes are smoothed and closer to the true population sizes than the raw counts.

```
# Define function to draw a graph to summarize results
graph.ssm <- function(ssm, N, y){
  fitted <- lower <- upper <- numeric()
  n.years <- length(y)
  for (i in 1:n.years) {
    fitted[i] <- mean(ssm$sims.list$N.est[,i])
    lower[i] <- quantile(ssm$sims.list$N.est[,i], 0.025)
    upper[i] <- quantile(ssm$sims.list$N.est[,i], 0.975)}
```



**FIGURE 5.3** Analysis of ibex population dynamics: trajectory of true population size (red), observed counts (black), and posterior means of estimated population size (blue; with 95% CRI shaded).

```

m1 <- min(c(y, fitted, N, lower))
m2 <- max(c(y, fitted, N, upper))
par(mar = c(4.5, 4, 1, 1), cex = 1.2)
plot(0, 0, ylim = c(m1, m2), xlim = c(0.5, n.years), ylab = "Population
size", xlab = "Year", las = 1, col = "black", type = "l", lwd = 2,
frame = FALSE, axes = FALSE)
axis(2, las = 1)
axis(1, at = seq(0, n.years, 5), labels = seq(0, n.years, 5))
axis(1, at = 0:n.years, labels = rep("", n.years + 1), tcl = -0.25)
polygon(x = c(1:n.years, n.years:1), y = c(lower, upper[n.years:1]),
col = "gray90", border = "gray90")
points(N, type = "l", col = "red", lwd = 2)
points(y, type = "l", col = "black", lwd = 2)
points(fitted, type = "l", col = "blue", lwd = 2)
legend(x = 1, y = m2, legend = c("True", "Observed", "Estimated"),
lty = c(1, 1, 1), lwd = c(2, 2, 2), col = c("red", "black", "blue"),
bty = "n", cex = 1)
}

# Execute function: Produce figure
graph.ssm(ssm, N, y)

```

### 5.3 SYSTEMATIC BIAS IN THE OBSERVATION PROCESS

In the previous example we have seen that the model performs well in the presence of “random” observation errors. By this we mean that on average, false positives (i.e., double-counting) and false negatives (i.e., nondetection)

cancel out. In this sense, the state-space model is able to “correct” noisy counts for observation error. However, the model cannot recover unbiased estimates of true population size, when false negative and positive errors don’t cancel out on average. With the data considered in this chapter, there is no direct information about the observation process, that is, about detection probability or double-counting rates. Thus, in the absence of false-positive errors, the “state” modeled in the state-space models of this chapter is the product of population size  $N$  and detection probability  $p$ , and not population size  $N$ , as one might think. We illustrate this next.

We start with a simple example where we assume that the ibex population remains stable at 50 individuals over 25 years. Further, we assume that detection probability is constant at 0.7, thus on average we only count 70% of the population. Each individual can be seen or missed, so the annual count is a binomial random variable, and we write  $y_t \sim \text{Bin}(N_t, p)$ . We start by defining the true population sizes.

```
n.years <- 25 # Number of years
N <- rep(50, n.years)
```

Then, we simulate the counts using the binomial distribution and inspect the numbers.

```
p <- 0.7
y <- numeric(n.years)
for (t in 1:n.years) {
  y[t] <- rbinom(1, N[t], p)
}
y
[1] 34 34 35 38 29 37 ... 27 35 33 32
```

Clearly, the observed counts are smaller than the true population size (what a surprise!). More importantly, we also notice that the counts vary, although both the true population size and the detection probability were constant across time. As we have seen in Section 1.3, this variation is the binomial sampling variation. The magnitude of this variance is known from statistical theory to be on average  $Np(1 - p) = 10.5$ .

We next use WinBUGS to fit to these data the same state-space model as in Section 5.2. Thus, we partition the observed variation in the counts into process variation and observation error and use a normal approximation for the latter. In large samples, we would expect the estimate of the process variance to be close to zero and that for the observation error to be close to 10.5.

```
# Bundle data
bugs.data <- list(y = y, T = n.years)
```

```

# Initial values
inits <- function(){list(sigma.proc = runif(1, 0, 5), mean.lambda =
runif(1, 0.1, 2), sigma.obs = runif(1, 0, 10), N.est = c(runif(1, 30,
60), rep(NA, (n.years-1))))}

# Parameters monitored
parameters <- c("lambda", "mean.lambda", "sigma2.obs", "sigma2.proc",
"N.est")

# MCMC settings
ni <- 25000
nt <- 3
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
ssm <- bugs(bugs.data, inits, parameters, "ssm.bug", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

# Summarize posteriors
print(ssm, digits = 3)

      mean     sd   2.5%    25%    50%    75%   97.5%   Rhat n.eff
[...]
mean.lambda  0.999  0.009  0.981  0.994  0.998  1.002  1.020  1.015  4800
sigma2.obs   15.219  5.644  7.157  11.370  14.170  18.020  28.500  1.005  660
sigma2.proc   0.002  0.003  0.000  0.000  0.001  0.002  0.011  1.012  210
[...]

```

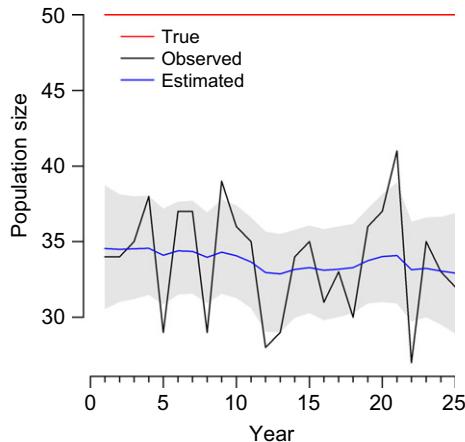
As expected, the process variance is estimated to be close to zero and the estimate of the observation variance is large. As a result of sampling variation, the latter estimate is of course not equal to 10.5, although a 95% CRI covers the expected value. With larger population sizes and averaged over many simulation replicates, we would expect to be right on target (see also Section 1.5). The estimate of the average population growth rate is essentially equal to 1, which is also what we would expect from a constant population. Now, let's compare the estimated and true population sizes, as well as the counts (Fig. 5.4).

```

# Produce figure
graph.ssm(ssm, N, y)

```

Figure 5.4 shows that the estimated population sizes are consistently lower than the true sizes, but the temporal pattern of the estimated population sizes matches quite well to that of the true population sizes. Thus, the model is able to correct for the temporal variation of the observation process induced by the binomial nature of the counts. However, the model cannot fully correct for the detection error; the population estimates are all well below the true population size. The mean (over time) of the estimated population sizes is 33.7, which is close to what we would expect,  $N_p$  ( $50 \cdot 0.7 = 35$ ). Hence, we obtain an improved estimate of the population index  $N_p$  with the



**FIGURE 5.4** Effects of systematic observation errors on the state-space model: estimated states represent an unbiased population index. True (red), observed (black), and estimated trajectory of population size (blue; with 95% CRI shaded).

state-space model, by eliminating the effects of annual observation errors, but we are unable to estimate true population size. We feel that the nature of the correction for the observation error in the kind of state-space models considered in this chapter is sometimes misunderstood among ecologists.

Next we conduct a simulation where detection probability is not constant over time, but instead has a positive trend that is linear on the logit scale. What will our simple state-space model tell us about the true trend of the ibex population in this case?

We start by defining the true population sizes.

```
n.years <- 25 # Number of years
N <- rep(50, n.years)
```

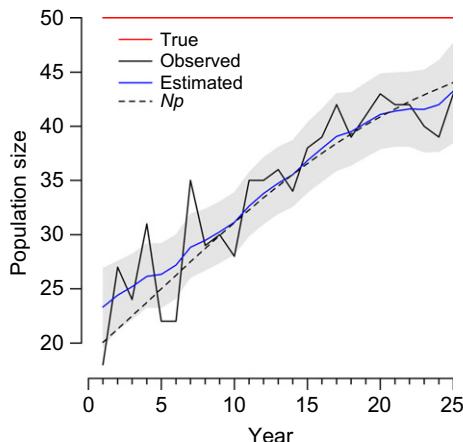
Then we simulate the counts using the binomial distribution.

```
lp <- -0.5 + 0.1*(1:n.years)      # Increasing trend of logit p
p <- plogis(lp)
y <- numeric(n.years)
for (t in 1:n.years){
  y[t] <- rbinom(1, N[t], p[t])
}
```

We analyze the data with the same model as before:

```
# Bundle data
bugs.data <- list(y=y, T=n.years)

# Call WinBUGS from R (BRT <1 min)
ssm <- bugs(bugs.data, inits, parameters, "ssm.bug", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())
```



**FIGURE 5.5** Effects of temporal change in systematic observation errors on the state-space model: estimated states represent a biased population index. True (red), observed (black), and estimated population size (blue) along with a 95% CRI (shading). The dashed line shows the expectation of the counts under binomial sampling.

The comparison among true, observed, and estimated population size shows that the model succeeds in getting rid of the random sampling variation in the counts that is resulting from binomial sampling (Fig. 5.5). However, the model can't correct the estimated population trajectory for the shift imposed by the deterministic trend in the observation process. To clarify this, we also calculated the expected value of the counts as  $N_p$  and added this to the plot. Clearly, the estimated population size under the state-space model follows quite closely to this expected value ( $N_p$ ). In other words, our simple state-space model was not able to correct for a systematic pattern in detection, and consequently we would erroneously infer a population increase if we trusted the model result in this case.

```
# Produce figure
graph.ssm(ssm, N, y)
points(N*p, col = "black", type = "l", lwd = 2, lty = 2)
legend(x = 1, y = 45.5, legend = "Np", lwd = 2, col = "black", lty = 2,
      bty = "n")
```

These examples illustrate three important points about the state-space models in this chapter:

1. They yield unbiased estimates of population size only if false-negative (detection probability) and false-positive observations (double-counting) cancel out on average. Counts are then unbiased on average but there is sampling variation, and this is corrected for by the model.
2. They produce unbiased estimates of population indices (i.e., of  $N_p$ ) if detection probability is  $<1$  and has no pattern over time. In that case,

- for the example of imperfect detection (and no false positives), the state-space model corrects for binomial sampling variation.
3. They do not yield unbiased estimates of population size nor of indices if there are temporal patterns either in detection probability or in the false-positive rates (unless, of course, the two cancel out).

## 5.4 REAL EXAMPLE: HOUSE MARTIN POPULATION COUNTS IN THE VILLAGE OF MAGDEN

The population of house martins (Fig. 5.6) in Magden (a small village in Northern Switzerland where MS grew up) has been surveyed by Reto Freuler since 1990. In initial years, Freuler counted all occupied nests himself, but later he sent questionnaires to house owners with known house martin nests asking for the number of occupied nests to be reported to him. The goal of the analysis is to estimate the average stochastic population growth rate, the process variance and to predict population sizes until 2015 with an assessment of uncertainty. Furthermore, we want to estimate the probability that the population size in 2015 is lower than in 2009, the year with the most recent data. In contrast to the earlier examples, we now adopt the exponential growth model on the log scale because it is more appropriate for stochastic environments (Lande et al., 2003).



**FIGURE 5.6** House martin (*Delichon urbica*), Finland, 2005 (Photograph by T. Muukkonen).

The state-process model, now, becomes  $\log(N_{t+1}) = \log(N_t) + r_t$ , with  $r_t \sim N(\bar{r}, \sigma_r^2)$ ; here,  $r_t$  is the stochastic population growth rate.

We use a similar state-space model to the one introduced earlier, with the mentioned exception (log scale). We also modify the priors for the initial population size, for the mean growth rate, and for the standard deviation of the state and the observation processes. Sometimes, it can be hard to fit state-space models in WinBUGS, which may fall in a trap or fail to start updating. Often, this has to do with the prior or the initial value for the first-year population size. It is advisable to choose a prior with mean equal to the first-year count and a relatively small variance. The range of the initial value for the first population size should also be relatively small. Although we specifically aim to predict population sizes in the future, the model does not need any changes for this purpose.

```
# Specify model in BUGS language
sink("ssm.bug")
cat("
model {

# Priors and constraints
logN.est[1] ~ dnorm(5.6, 0.01) # Prior for initial population size
mean.r ~ dnorm(1, 0.001) # Prior for mean growth rate
sigma.proc ~ dunif(0, 1) # Prior for sd of state process

sigma2.proc <- pow(sigma.proc, 2)
tau.proc <- pow(sigma.proc, -2)
sigma.obs ~ dunif(0, 1) # Prior for sd of obs. process
sigma2.obs <- pow(sigma.obs, 2)
tau.obs <- pow(sigma.obs, -2)

# Likelihood
# State process
for (t in 1:(T-1)) {
  r[t] ~ dnorm(mean.r, tau.proc)
  logN.est[t+1] <- logN.est[t] + r[t]
}

# Observation process
for (t in 1:T) {
  y[t] ~ dnorm(logN.est[t], tau.obs)
}

# Population sizes on real scale
for (t in 1:T) {
  N.est[t] <- exp(logN.est[t])
}
}

", fill=TRUE)
sink()
```

Next, we load the data. They consist of a vector with the number of observed breeding pairs in each year. Because we want to predict future

population sizes, we add NAs in the data. Why? Well, we saw in Chapter 2 that Bayesians treat all types of unknown quantities in the same way, be they predictions, missing values, or parameters. Hence, it is enough to specify some additional years without observations (regardless of whether they occur within the data series, or only at the end) with NA observations, and include them in the loop for calculating the likelihood. They are then updated (estimated) as part of the model fitting. Here, we would like to predict population size in year 2015, and so we need to add six NAs. Another way to achieve the same would be to extend the loop of the state-process to cover the additional years, and to leave the data as they are (i.e., without NAs).

```
# House martin population data from Magden
pyears <- 6 # Number of future years with predictions
hm <- c(271, 261, 309, 318, 231, 216, 208, 226, 195, 226, 233, 209, 226,
       192, 191, 225, 245, 205, 191, 174, rep(NA, pyyears))
year <- 1990:(2009 + pyyears)
```

We then bundle the data, specify initial values, list parameters to be estimated, determine the MCMC settings, and run the model.

```
# Bundle data
bugs.data <- list(y = log(hm), T = length(year))

# Initial values
inits <- function(){list(sigma.proc = runif(1, 0, 1), mean.r = rnorm(1),
    sigma.obs = runif(1, 0, 1), logN.est = c(rnorm(1, 5.6, 0.1), rep(NA,
    (length(year)-1))))}

# Parameters monitored
parameters <- c("r", "mean.r", "sigma2.obs", "sigma2.proc", "N.est")

# MCMC settings
ni <- 200000
nt <- 6
nb <- 100000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
hm.ssm <- bugs(bugs.data, inits, parameters, "ssm.bug", n.chains = nc,
    n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
    bugs.dir, working.directory = getwd())
```

Convergence is satisfactory after 200,000 iterations after a 100,000 burnin. The least well-mixed parameters are, not surprisingly, the two variances. If WinBUGS does not run properly and produces an undefined real result, just start the run again. The inits function will produce a different set of starting values each time it is called.

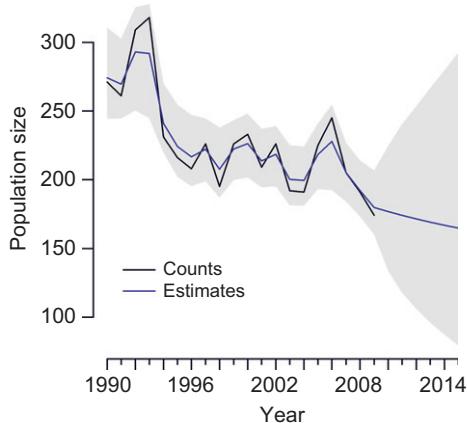
```
# Summarize posteriors
print(hm.ssm, digits = 2)

      mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
r[1] -0.02  0.06 -0.13 -0.05 -0.02  0.01  0.11 1.00 47000
[...]
r[25] -0.02  0.11 -0.26 -0.08 -0.02  0.04  0.21 1.00 27000
mean.r -0.02  0.03 -0.08 -0.04 -0.02 -0.01  0.03 1.00 39000
sigma2.obs 0.01  0.01  0.00  0.00  0.00  0.01  0.02 1.00  870
sigma2.proc 0.01  0.01  0.00  0.00  0.01  0.02  0.03 1.01  770
N.est[1] 274.10 15.88 244.50 265.20 272.30 282.10 310.40 1.00 7100
N.est[2] 269.51 14.50 244.60 260.30 267.20 277.70 302.50 1.00 4500
[...]
N.est[26] 164.84 54.42 79.43 131.50 159.30 188.60 292.30 1.00 44000
```

We plot counts and the posterior means of population sizes along with their credible intervals.

```
# Draw figure
fitted <- lower <- upper <- numeric()
year <- 1990:2015
n.years <- length(hm)
for (i in 1:n.years) {
  fitted[i] <- mean(hm.ssm$sims.list$N.est[,i])
  lower[i] <- quantile(hm.ssm$sims.list$N.est[,i], 0.025)
  upper[i] <- quantile(hm.ssm$sims.list$N.est[,i], 0.975)}
m1 <- min(c(fitted, hm, lower), na.rm = TRUE)
m2 <- max(c(fitted, hm, upper), na.rm = TRUE)
par(mar = c(4.5, 4, 1, 1))
plot(0, 0, ylim = c(m1, m2), xlim = c(1, n.years), ylab = "Population
  size", xlab = "Year", col = "black", type = "l", lwd = 2, axes = FALSE,
  frame = FALSE)
axis(2, las = 1)
axis(1, at = 1:n.years, labels = year)
polygon(x = c(1:n.years, n.years:1), y = c(lower, upper[n.years:1]),
  col = "gray90", border = "gray90")
points(hm, type = "l", col = "black", lwd = 2)
points(fitted, type = "l", col = "blue", lwd = 2)
legend(x = 1, y = 150, legend = c("Counts", "Estimates"), lty = c(1, 1),
  lwd = c(2, 2), col = c("black", "blue"), bty = "n", cex = 1)
```

Counts and estimated population sizes are quite close (Fig. 5.7), which is reflected in a small estimate of the observation variance. The overall trajectory of the house martin population is negative, but with important annual fluctuations. This can be seen in Fig. 5.7, but is apparently not reflected in the estimate of the process variance, which is only about double of the observation variance. Is there something wrong? No, there is nothing wrong; we just can't compare variances directly if they are measured around different means. A better way to compare variances is to calculate the coefficient of variation (CV), which is the ratio of the



**FIGURE 5.7** Counts (black) and estimated population size (blue) of house martins in Magden (with 95% CRI shaded).

standard deviation to the mean. The average population size over the 20 study years is about 230 pairs, thus the CV of the observation error is  $\frac{\sqrt{0.006}}{\log(230)} = 0.014$ . The CV of the process variation is  $\frac{\sqrt{0.012}}{0.022} = 4.98$ , and thus about 350 times larger than the CV of the observation error.

The projected population sizes after 2009 are based on the estimates of the population size in 2009, the average growth rate of the population, and the process variance. On average, we predict a steadily declining population, but the uncertainty in the estimated population trajectory is large (as reflected by the wide CRIs). This is mostly because of the large process variance, so it is difficult to make precise predictions. Typically, the uncertainty becomes larger the further we predict into the future. The predicted trajectory is the Bayesian analog of what Lande et al. (2003) call a population prediction interval (PPI). However, it is not based on the bootstrap and the usual approximations involved with maximum likelihood. Such a prediction with full uncertainty assessment is easy to obtain in a Bayesian framework of inference, but is more challenging in the frequentist arena.

We were also asking about the probability that the house martin population in 2015 would be smaller than in 2009. This quantity is again a derived variable and can easily be obtained from the MCMC output. We evaluate the proportion of MCMC samples of the estimated population size in 2015 that is smaller than in 2009. The probability that the population size in 2015 is smaller than in 2009 is 0.689 (code below), the probability that it would be larger is 0.311. Hence, a decline of the house martin population until 2015 is twice as likely as the inverse.

```
# Probability of N(2015) < N(2009)
mean(hm.ssm$sims.list$N.est [,26] < hm.ssm$mean$N.est [20])
```

## 5.5 SUMMARY AND OUTLOOK

We have introduced classical state-space models of population dynamics, which are used to analyze population counts and to partition the observed variation into a component due to process variation and another due to observation error. We have seen that these models can be useful in some situations, but it must be kept in mind that they usually do not provide an unbiased estimate of the true population size. Instead, they yield a smoothed estimate of a population index (i.e., of  $Np$ ), which may or may not be unbiased with respect to the true population trajectory. In the absence of temporal patterns in the observation error, the trajectories of the true population sizes and that of the estimated population index will be parallel. The smoothing occurs with respect to the assumed underlying biological processes, and is therefore more mechanistic and perhaps more realistic than the arbitrary smoothing of generalized additive models (GAM; King et al., 2010), which are often used to analyze population counts (Fewster et al., 2000).

We have considered only very simple state-space models to describe state and observation processes. The beauty of these models is that we can readily change them to be more realistic or in a way that specific hypotheses can be tested. An obvious extension is to use a density-dependent model such as the logistic, theta-logistic, or Gompertz model (e.g., Lande et al., 2003; Dennis et al., 2006). However, it appears that models with density-dependence have serious extrinsic identifiability problems. With increasing magnitude of the observation error, the parameters become harder to estimate, especially the parameter which quantifies the strength of density-dependence (Knape, 2008).

If our aim is to get unbiased estimates of the population size, the use of state-space models of the type featured in this chapter is not sufficient. In other words, there is no way around explicitly estimating detection probability. In Chapters 6 (for a single closed population), 10 (for a single open population), and 12 (for metapopulations), we will see how this can be done. This requires additional information to tease apart detection probability and the true levels of the state process (population size). However, when no such additional data are available about the detection process, the state-space models in this chapter represent probably one of the best frameworks for inference about population trajectories. In Chapters 7–10, we will use different, more mechanistic state-space models to estimate survival and other demographic quantities from capture–recapture, mark–recovery, and multistate capture–recapture data, and in Chapter 13 for metapopulation dynamics.

## 5.6 EXERCISES

1. Random variability in detection probability: quite often we cannot assume that detection probability is constant over time, for example, because of weather factors that affect the counts. Simulate and analyze

- data for a population, whose size remains constant at 50 individuals over (a) 25 years and (b) 50 years, but where the annual detection probability varies randomly in the interval between 0.3 and 0.7. Does the state-space model perform well in this situation?
2. Modeling of variance structures: in the house martin data set, we saw that from year  $t = 9$  onwards, a different data-collection protocol (questionnaires) was used. Adapt the model to account for possibly different observation errors in the two periods.
  3. Unstructured and dynamic hierarchical model for population counts: In Section 3.3.2, we encountered a different two-level hierarchical model for a single time series of population counts. What is the difference to a state-space model in this chapter? Fit the exponential population state-space model to the peregrine data from Section 3.3.2 and compare the inference about the population trajectory under the two models. In addition, construct a model with a linear trend in the population growth rate and another model with a linear trend in the observation error and fit them to the peregrine data.

# Estimation of the Size of a Closed Population from Capture–Recapture Data

## OUTLINE

6.1	Introduction	134
6.2	Generation and Analysis of Simulated Data with Data Augmentation	139
6.2.1	Introduction to Data Augmentation for the Simplest Case: Model $M_0$	139
6.2.2	Time Effects: Model $M_t$	145
6.2.3	Behavioral or Memory Effects: Model $M_b$	148
6.2.4	Individual (Random) Effects: The Heterogeneity Model $M_h$	150
6.2.5	Combined Effects: Model $M_{th}$	154
6.3	Analysis of a Real Data Set: Model $M_{tbh}$ for Species Richness Estimation	157
6.4	Capture–Recapture Models with Individual Covariates: Model $M_{t+x}$	162
6.4.1	Individual Covariate Model for Species Richness Estimation	163
6.4.2	Individual Covariate Model for Population Size Estimation	166
6.5	Summary and Outlook	169
6.6	Exercises	170

## 6.1 INTRODUCTION

In Chapter 5 (and partly in Chapter 4; see Section 4.4), we met models that attempt to distinguish between an ecological process and an observation process. This is important, since it allows for a more proper accounting of the uncertainty in a modeled system (Cressie et al., 2009) and is required to avoid bias in some important system descriptors, such as density dependence (Freckleton et al., 2006). However, at a closer look, these models do not fully deliver what they promise at first sight: after all, they typically do not have parameters with a clear ecological meaning. Or, put in another way, what exactly is the ecological relevance of an “expected count” (Bob Dorazio, pers. comm.)? After all, we would like to describe a population in terms of its size, not in terms of the field method that we apply to produce a count of animals or plants.

In this Chapter, we introduce a different class of models that achieve a proper accounting for the processes that give rise to observed data, in terms of an ecological and of an observation process: closed-population capture–recapture models. Capture–recapture models in general are a very large class of models that have become increasingly used in ecological applications of statistical modeling (Seber, 1982; Borchers et al., 2002; Williams et al., 2002; Royle and Dorazio, 2008; King et al., 2010). In principle, they all boil down to estimation of detection probability, which provides the link between what we observe and the true population parameters, such as population size or survival probability. Detection probability can be estimated from repeated encounters of individually marked animals or plants. Historically, encountering meant capturing an animal and hence, the associated statistical methods have come to be called capture–recapture, or mark-recapture, models. However, the essential feature of these models is that they require as data repeated *observations of individually recognizable units*, such as individuals, occupied sites, or species, to estimate population size, number of occupied patches, or species richness, respectively. Individual recognition may well be possible without physical capture.

Almost all of the remainder of this book will be dedicated to models for inference about populations which contain an explicit parameter representing detection probability. These models describe the observation process explicitly and include capture–recapture models for marked individuals and other methods for unmarked “individuals” as well; see the metapopulation estimation models in Chapters 12 and 13. There is a single exception in Chapter 11, where we partly fall back on the modeling of Chapters 4 and 5. The models in Chapter 6 deal with repeated surveys from a single population. Surveys are conducted over a period that is sufficiently short so that the population can be assumed “closed” to

numerical changes, that is, no recruitment, mortality, and dispersal occur. Most of the methods in later chapters relax that assumption, at the expense of not being able to estimate abundance (but see Chapter 10 and also Chapters 12 and 13).

Assessing the size of a closed population ( $N$ ), or analogously, its density ( $D$ ), appears simple at first sight; simply tally up all individuals ( $N$ ) that live on a piece of land ( $A$ ). However, in reality, assessing  $N$  is fairly challenging because of at least two reasons: imperfect detection ( $p$ ) and the problem of delimiting the size of the study area ( $A$ ).

First, imperfect detection is a hallmark of all studies of wild organisms, be they plants (Kéry and Gregg, 2003; Kéry, 2004; Kéry et al., 2005a) or animals (Williams et al., 2002; Nichols et al., 2009). In most cases and under most situations, not every individual present in a study area will be seen and hence, counts ( $C$ ) will typically be an underestimate of true population size ( $N$ ). Over several decades, a huge number of protocols and associated models have been developed to cope with this challenge and to “correct” the observed counts for detection error induced by  $p < 1$ . Conceptually, all derive some estimates of the probability to detect a member of  $N$ , that is, of detection probability  $p$ , and then apply the commonsense or canonical estimator for  $N$ , which is  $\hat{N} = C/\hat{p}$ , where hats denote estimates.

One of the main differences between these methods is in how they derive their estimate of  $p$ . Classical capture–recapture models use repeated observations of recognizable individuals and derive information about  $p$  from the pattern of detection and nondetection of each marked individual. Capture–recapture methods are huge in scope, and there is a tremendous variety of protocols and associated models (Borchers et al., 2002; Williams et al., 2002; Royle and Dorazio, 2008). Distance sampling methods derive information about  $p$  from the distribution of the distances between an observer (or some detection device) and the detected animals or objects sampled (Buckland et al., 2001). Repeated counts methods (Royle, 2004c) do not require individual identification of the objects counted; they are a third and very useful protocol (see Chapter 12).

The second challenge in assessing  $N$  is that we need to decide how to delimit the study area or put another way, how should we define the area with which our population is associated or simply, what is  $A$ ? Obviously, this is trivial for actual islands or islands of habitat such as mountain tops or desert oases. However, in most cases, objectively delimiting the area that is used by a population is difficult or may be downright impossible without making some arbitrary decisions.

One reason it may be difficult to determine the area on which a population lives is movement of animals within their home range. Any defined area will then sometimes contain individuals who have the center of gravity of their entire activity range outside of that perimeter, and who, therefore,

in a sense, do not belong to the focal population but can nevertheless be seen within a study area. Similarly, some individuals at the edge of the study area, who do have the majority of their activity range within the nominal area associated with the desired population, may sometimes move beyond that perimeter and therefore be absent and undetectable within the perimeter. But how should an individual at the border of the study area be counted? As a fractional individual only? Or full, if its home-range center is inside and not at all, when it is outside the boundary of the nominal study area? Dealing with this sort of *temporary emigration* (as it is called in the capture–recapture literature; Kendall et al., 1997; Kendall, 1999) is almost always a challenge in population size estimation (although neither in the case of distance sampling nor in spatially explicit capture–recapture models). Conceptually, it is related to the problem of separating permanent emigration from mortality probability in the Cormack–Jolly–Seber model (Chapter 7). Spatially explicit capture–recapture (SECR) models (e.g., Efford, 2004; Borchers and Efford, 2008; Royle and Young, 2008) should be applied in this context because they effectively deal with the challenge of determining the size of  $A$ .

In this chapter, we do not describe spatial capture–recapture models but instead assume that we are able to delimit a piece of land with which some unknown number  $N$  of animals or plants are associated, and we want to estimate that number  $N$  by “correcting” our count by our estimate of detection probability  $p$ . One important further assumption is that there are no misclassifications, that is, no individual is mistaken for another. This is the classical setting of a capture–recapture study, where we need repeated surveys to obtain so-called detection histories, strings of ones and zeroes, that denote whether an individual was detected or not during any given repeat survey (also called an occasion). Classical capture–recapture models consist of modeling patterns in  $p$  and obtain an estimate of  $N$  through the canonical estimator  $C/\hat{p}$ .

Otis et al. (1978) provided a useful catalog of possible patterns in detection probability in the context of closed-population size estimation. They distinguish between three effects: individual effects, time effects, and behavioral response effects. Individual and time effects are the differences among individuals and those among capture occasions. Behavioral response denotes a situation wherein detection probability at an occasion depends on whether an individual was detected in a previous occasion. The term comes from live trapping of animals, where such a nonindependence of detection probability within an individual detection history is due to the individual changing its behavior in response to a capture event. If trapping leads to an increased detection probability at a later occasion, one talks about “trap-happiness”; the converse is called “trap-shyness”. However, patterns akin to a trap response may arise also as a result of quite different mechanisms, for instance, if an observer

**TABLE 6.1** Parameters for Detection Probability  $p_{it}$  in a Hypothetical Study with Three Individuals and Four Capture Occasions.

Individual	Occasion 1	Occasion 2	Occasion 3	Occasion 4
1	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$
2	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$
3	$p_{3,1}$	$p_{3,2}$	$p_{3,3}$	$p_{3,4}$

Note: The models  $M_0$ ,  $M_t$ ,  $M_b$ ,  $M_h$  (see text) differ in the assumptions that they make about how cells are related among rows, columns, and within a row.

remembers the location of a species in a study of species richness (e.g., an owl in a hollow tree) and is more likely to detect a species that he/she has detected previously. In addition, individual heterogeneity in detection probability may appear like trap-happiness (Kéry et al., 2006).

The three main classes of models in Otis et al. (1978) are known as models  $M_h$  (h for individual heterogeneity),  $M_t$  (t for time), and  $M_b$  (b for behavioral response). These effects can be visualized in a cross-classification of individuals and occasion such as in Table 6.1 for a toy study of three individuals and four occasions. Model  $M_h$  accounts for row effects (i.e., individuals differ), model  $M_t$  accounts for column effects (occasions differ), and model  $M_b$  accounts for within-row effects, that is, within-individual capture-history dependence.

Otis et al. (1978) defined eight models using these three effects: Model  $M_0$ , which assumes that  $p$  is constant across all individuals and times; models  $M_h$ ,  $M_t$ , and  $M_b$  as defined earlier; and four models with two-way and three-way combinations of effects:  $M_{th}$ ,  $M_{bh}$ ,  $M_{tb}$ , and  $M_{tbeh}$ . Estimators for most of these models were implemented in the grandfather of population size estimation software, CAPTURE (Rexstad and Burnham, 1991). Many of these methods have now been superseded by more efficient (e.g., likelihood-based) estimators, and many of those are available in program MARK (White and Burnham, 1999). Indeed, over the last 1–2 decades there has been a proliferation of new methods for population size estimation, see Borchers et al. (2002), Williams et al. (2002) and also Chapter 14 by Paul Lukacs in the *Gentle Introduction to program MARK* (<http://www.phidot.org/software/mark/docs/book/>). The Otis et al. (1978) catalog of effects remains useful because it clarifies our thinking about possible patterns in such collections of parameters in a cross-classification of individual by time.

Three things must be remembered however. First, what used to be called model t, b, or h is in fact not a single model, but in reality a whole family of models: there are various ways in which temporal, behavioral, or individual effects may be modeled. For instance, we may want to model a permanent trap response, such as is the tradition in population

size estimation, or we may want to model an immediate trap response, as is customary in Cormack–Jolly–Seber models (see Section 7.8; Appendix 2.2; Pradel, 1993). Moreover, there are many different ways to model individual heterogeneity, for instance, by finite or continuous mixture distributions, and for continuous mixtures, by different parametric forms (Pledger, 2000; Dorazio and Royle, 2003; Link, 2003). So, it is really not enough to say “we used model  $M_b$ ” or “we used model  $M_h$ ”; rather, we need to say (and know) which model  $M_b$  or which model  $M_h$  we are using.

Second, as there is not a single model  $M_b$  for instance, there is not a single model  $M_{tb}$  either. When two effects are present in a model, we can combine them in an additive or in an interactive way. So, now that we are not constrained anymore to the rigid set of models for population size estimation from the old CAPTURE days, we must also say (and know) how to combine different effects in a model.

Third, all models in the Otis et al. (1978) classification represent “group” effects, where we batch the parameters in Table 6.1 in some way and estimate differences among these batches. However, once we recognize how these models can be specified as GLMs or GLMMs, we immediately have much more modeling freedom. We can do all this modeling on the scale of the logit-transformed detection probability ( $p$ ) and then combine group effects and continuous covariate effects in a linear model of the usual form, for instance:

$$\text{logit}(p_{i,j}) = \alpha_i + \beta_j + \delta * F_{i,j} + \gamma * X_{i,j},$$

$$\text{with } \alpha_i \sim \text{Normal}(\mu_\alpha, \sigma_\alpha^2).$$

For illustration, in this model for detection probability of individual  $i$  during occasion  $j$ , we specify random individual effects  $\alpha_i$ , fixed time effects  $\beta_j$ , an effect  $\delta$  that depends on whether an animal was captured before or not (this information is contained in the indicator variable  $F$ ), and an effect  $\gamma$  of another covariate  $X$ . In any practical application, we may be far from being able to estimate all these quantities, there may be confounding and not enough data etc. Also, one of the fixed time effects ( $\beta_j$ ) would have to be constrained to zero to avoid overparameterization.

So, it is useful to think about all these models in a GLM way. But in addition, we find it useful to be able to relate a model that is written in this way back to the Otis et al. (1978) classification. For instance, our model above would represent a model  $M_{tbh}$  with an added covariate  $X$  and where all effects are additive rather than interactive (so, we may also want to write that model as  $M_{t+b+h+X}$ ). WinBUGS gives us full modeling freedom to fit such custom models because its model definition language is so apt to free the modeler in us (Kéry, 2010).

It is worth noting that the linear models concept (i.e., the design matrix) underlying capture–recapture models when viewed as GLM are not only

important for population size estimation. Far from that: the modeling of time effects (e.g., fixed or random or as a function of covariates), the modeling of “behavioral” effects and the modeling of individual effects are elements that we will see as part of quite different models over and over again. Hence, a full understanding of the material in this chapter will be a great help for your modeling in WinBUGS for a vast range of models. Essentially the same topics arise in other such collections of parameters, for instance, in the modeling of survival probability in a cross-classification of individual and time in the Cormack–Jolly–Seber model (Chapter 7) or of detection probability in a cross-classification of site and time in a metapopulation model for abundance (Chapter 12) or occurrence (Chapter 13). We believe that seeing these connections will be very helpful for obtaining a more synthetic understanding of capture–recapture models. Hence, the way in which we structure such tables of parameters, which is implied by the cross-classification of individuals and time in this chapter is relevant for the modeling of tables of parameters in many other inferential situations as well.

This chapter is the place to describe for the first time data augmentation (Tanner and Wong, 1987), introduced in the context of capture–recapture models by (Royle et al., 2007a); see also Royle and Dorazio (2011) and Section 10.3. Data augmentation greatly simplifies inference for a vast range of models and makes them amenable to simple fitting in WinBUGS. In particular, data augmentation offers an extremely flexible way to model patterns of detection probability in closed populations. We first introduce data augmentation with the simplest possible closed-capture model,  $M_0$ . Then, we will look into a few simple examples of closed-capture models with simulated data:  $M_t$ ,  $M_b$ ,  $M_h$ , and  $M_{th}$ . With model  $M_t$ , we will see again how one simply changes from fixed effects to random effects, and how transparent this is when the model is described in the BUGS language. Finally, we will analyze a real data set under models  $M_{tbh}$  and  $M_{tbh+X}$ , where “population size” is represented by species richness. In the latter model, we provide an illustration of individual covariate modeling, which is greatly simplified when using data augmentation (Royle, 2009).

## **6.2 GENERATION AND ANALYSIS OF SIMULATED DATA WITH DATA AUGMENTATION**

### **6.2.1 Introduction to Data Augmentation for the Simplest Case: Model $M_0$**

The simplest possible model for inference about the size of a single population is model  $M_0$  (Otis et al., 1978), where detection probability is assumed constant over the two dimensions of the detection parameter

table, that is, over individuals and over time periods (Table 6.1). First, we write a function that simulates capture data under model  $M_0$ . The three arguments are  $N$  (population size),  $p$  (detection probability), and  $T$  (the number of sampling occasions).  $T$  is assumed constant for all individuals, but this is for pure convenience. In the analysis of the real data, we will see how we deal with the case where  $T$  is not the same across all individuals.

```
# Define function to simulate data under M0
data.fn <- function(N = 100, p = 0.5, T = 3) {
  yfull <- yobs <- array(NA, dim = c(N, T))
  for (j in 1:T) {
    yfull[, j] <- rbinom(n = N, size = 1, prob = p)
  }
  ever.detected <- apply(yfull, 1, max)
  C <- sum(ever.detected)
  yobs <- yfull[ever.detected == 1,]
  cat(C, "out of ", N, "animals present were detected.\n")
  return(list(N = N, p = p, C = C, T = T, yfull = yfull, yobs = yobs))
}
```

We create one realization of the stochastic process represented by sampling a population of size  $N$ . This gives us our data set.

```
data <- data.fn()
```

On execution of that function, we get a list of R objects. We can have a look at them by typing the name of the object, `data`, or we can get a summary of them by doing this:

```
str(data)
> str(data)
List of 6
 $ N      : num 100
 $ p      : num 0.5
 $ C      : num 87
 $ T      : num 3
 $ yfull: num [1:100, 1:3] 0 0 0 1 1 0 0 1 1 0 ...
 $ yobs  : num [1:87, 1:3] 0 0 1 1 0 0 1 1 0 1 ...
```

Here, `yfull` is the full capture-history matrix of all  $N$  animals, including those that were never captured and which have an all-zero capture-history. Evidently, this is not what we ever get to observe. The observed data are called `yobs`, and they have  $C$  rows only ( $C \leq N$ ), corresponding to the observed count of animals. We will apply a model to use these data to make an inference based on the rules of probability, of how great  $N$  might likely be.

When modeling population size or other parameters in a model that contains population size using Bayesian MCMC techniques, one formidable

technical challenge is that the dimension of the parameter vector for  $N$  may change at every iteration of the MCMC algorithm. An ingenious solution is a technique called *data augmentation* (Tanner and Wong, 1987), introduced in the context of capture–recapture models by Royle et al. (2007a); see also Royle and Dorazio (2011) and Section 10.3. Parameter-expanded data augmentation (PX-DA), as it is more accurately called, consists of two things: (1) adding an arbitrary number of zeroes to the data set and (2) analyzing a reparameterized version of the original model. Essentially, it converts the closed-population model into an occupancy model (see Chapter 13) and turns the problem of estimating abundance  $N$  into that of estimating occupancy ( $\psi$ ). Data augmentation may be simple to do in practice, but it has far-reaching consequences: it greatly simplifies the fitting of a large range of capture–recapture type models (Royle et al., 2007b; Royle and Dorazio, 2008).

Data augmentation consists of augmenting a data set by adding a large number of “potential”, unobserved individuals, with all zero-only encounter histories. The augmented data set has dimension  $M$  by  $T$ , where  $M \gg N$  (remember,  $N$  is the unknown population size, and  $T$  is the number of sampling occasions). To this augmented data set we fit a zero-inflated version of the model we would fit if  $N$  were known. To do this, we add to the model a binary indicator variable, say  $z$ , which is an indicator for whether a row in the augmented data matrix represents a “real” individual, or one that does not exist in practice. These indicators are given a Bernoulli prior distribution, and the parameter of that distribution, say  $\Omega$ , is estimable from the data.  $\Omega$  may be called the inclusion probability, since it is the probability with which a member of the augmented data set,  $M$ , is included in the population of size  $N$ .

Data augmentation translates the problem of estimating  $N$  into the equivalent problem of estimating  $\Omega$ , since the expectation of  $N$  is equal to  $M\Omega$ . Formally, data augmentation induces for  $N$  a discrete uniform prior on the interval  $(0, M)$ . In words, we make our analysis under the formal prior assumption “We do not know how big  $N$  is, but it could be any integer number between 0 and  $M$ , the size of the augmented data set, with equal probability”.

Although simple in reality, our experience suggests that the idea of data augmentation takes some time to sink in. So, here is a numerical example that summarizes what we have just said. Assume that we want to estimate the size of a population with  $N = 100$ . Of course, that there are 100 individuals is unknown to us because we observe only  $C = 87$  of them. So, instead of analyzing the observed data set of size 87 and estimating detection probability  $p$  and population size through some variant of the canonical estimator  $\hat{N} = C/\hat{p}$ , we add to the observed detection history matrix, say, 150 rows consisting of all zeroes. This brings the size of the augmented data set to  $M = 237$ . This is equivalent to us saying that  $N$  could really be anywhere between 0 and 237. We then fit an occupancy model (see Chapter 13) to

these data, where we estimate detection probability  $p$  and the inclusion probability  $\Omega$  and obtain the estimate of  $N$  as a derived parameter by summing up the latent indicators  $z$  (the expectation of  $N$  is  $M\Omega$ ). So, let us see now how this works in practice for the data set previously generated.

```
# Augment data set by 150 potential individuals
nz <- 150
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T) ) )

# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
p ~ dunif(0, 1)

# Likelihood
for (i in 1:M) {
  z[i] ~ dbern(omega)           # Inclusion indicators
  for (j in 1:T) {
    yaug[i,j] ~ dbern(p.eff[i,j])
    p.eff[i,j] <- z[i] * p      # Can only be detected if z=1
    } #j
  } #i

# Derived quantities
N <- sum(z[])
}
", fill = TRUE)
sink()
```

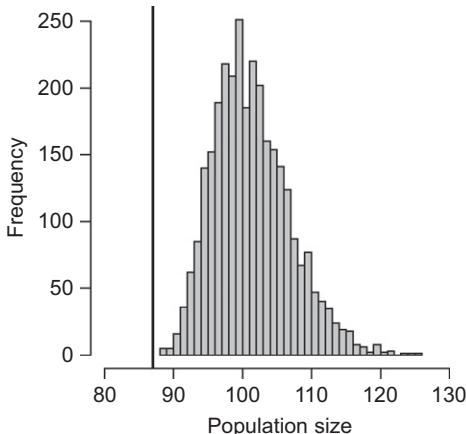
We started this chapter claiming that capture–recapture models achieve a proper separation of the ecological and the observation process, so where can we see this in the model? In fact, the first Bernoulli distribution governing the inclusion probabilities is equivalent to the ecological process (which creates “real” and nonexistent individuals), whereas the second represents the observation process.

We present the analysis in the usual format by preparing all the ingredients that `bugs()` require to instruct WinBUGS and then letting WinBUGS run. We estimate  $N$  at 101.8 with 95% CRI of 93–114 (Fig. 6.1), which comfortingly includes the truth of 100.

```
# Bundle data
win.data <- list(yaug = yaug, M = nrow(yaug), T = ncol(yaug))

# Initial values
inits <- function() list(z = rep(1, nrow(yaug)), p = runif(1, 0, 1))

# Parameters monitored
params <- c("N", "p", "omega")
```



**FIGURE 6.1** Posterior distribution of population size  $N$  (black line: observed number of animals).

```

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
            n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
            bugs.dir, working.directory=getwd())

# Summarize posteriors
print(out, dig=3)
hist(out$sims.list$N, nclass=50, col="gray", main="", xlab =
    "Population size", las=1, xlim=c(80, 150))
abline(v = data$C, lwd=3)
[...]
      mean     sd   2.5%    25%    50%    75%   97.5%   Rhat n.eff
N     101.803 5.547 93.000 98.000 101.000 105.000 114.000 1.003   940
p      0.479 0.038  0.404   0.453   0.480   0.505   0.555 1.002  1200
omega  0.430 0.039  0.356   0.404   0.428   0.456   0.507 1.001  3000
[...]

```

No doubt some of you will think that data augmentation is some sort of voodoo. Yet, it is not. To convince yourself that the number of zeroes added does not affect the estimates of detection probability ( $p$ ) and population size ( $N$ ), we repeat the analysis with the following numbers of all-zero detection histories added (i.e., values of  $\text{nz}$ ): 5, 150, and 1500. For each case, we will inspect the estimates for  $N$ ,  $p$ , and  $\Omega$ , along with their

uncertainty. To help understand data augmentation and this comparison, we also plot the posterior for  $N$  each time:

```

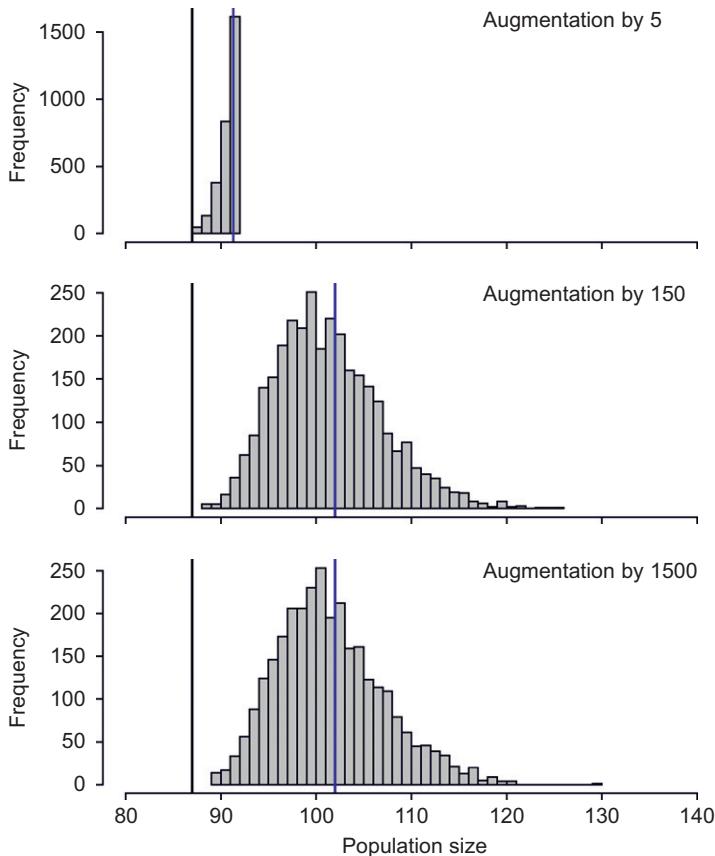
nz <- 5
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T)))
win.data <- list(yaug=yaug, M=dim(yaug)[1], T=dim(yaug)[2])
out5 <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
             n.thin=nt, n.iter=ni, n.burnin=nb, debug=FALSE, bugs.directory=
               bugs.dir, working.directory=getwd())
print(out5, dig=3)
par(mfrow=c(3, 1))
hist(out5$sims.list$N, nclass=30, col="gray", main="Augmentation
      by 5", xlab="Population size", las=1, xlim=c(80, 140))
abline(v=data$C, col="black", lwd=5)
abline(v=mean(out5$sims.list$N), col="blue", lwd=5)

nz <- 150
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T)))
win.data <- list(yaug=yaug, M=dim(yaug)[1], T=dim(yaug)[2])
out150 <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
                n.thin=nt, n.iter=ni, n.burnin=nb, debug=FALSE, bugs.directory=
                  bugs.dir, working.directory=getwd())
print(out150, dig=3)
hist(out150$sims.list$N, nclass=30, col="gray", main="Augmentation by
      150", xlab="Population size", las=1, xlim=c(80, 140))
abline(v=data$C, col="black", lwd=5)
abline(v=mean(out150$sims.list$N), col="blue", lwd=5)

nz <- 1500
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T)))
win.data <- list(yaug=yaug, M=dim(yaug)[1], T=dim(yaug)[2])
out1500 <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
                 n.thin=nt, n.iter=ni, n.burnin=nb, debug=FALSE, bugs.directory=
                   bugs.dir, working.directory=getwd())
print(out1500, dig=3)
hist(out1500$sims.list$N, nclass=30, col="gray", main=
      "Augmentation by 1500", xlab="Population size", las=1, xlim=
      c(80, 140))
abline(v=data$C, col="black", lwd=5)
abline(v=mean(out1500$sims.list$N), col="blue", lwd=5)

```

This exercise shows that provided you have augmented the data set enough (Fig. 6.2, central and bottom panels), data augmentation does not have any effect on the estimates, but simply on efficiency: more data augmentation is more costly in terms of computation time. But up to Monte Carlo error, the estimates and their standard errors remain the same (up to Monte Carlo error, as you can see in the posterior summaries). And how do you diagnose not enough data augmentation? Simple: just look at the posterior distribution of  $N$ : if it is truncated on the right by your choice of  $M$  (as it is in the top panel), you need to repeat the analysis with more zeroes added, that is, larger  $M$ .



**FIGURE 6.2** Posterior distributions of population size  $N$  for the same data set but under different degrees of data augmentation (black line: observed number of animals, blue line: posterior mean).

### 6.2.2 Time Effects: Model $M_t$

Another model assumes that detection probability  $p$  varies by occasion, perhaps because of weather conditions or because different traps or detection devices were used. The simultaneous use of different detection methods (e.g., trap types or observers) during a single occasion can be treated exactly as time effects in capture–recapture modeling. This model is called model  $M_t$  by Otis et al. (1978).

```
# Define function to simulate data under Mt
data.fn <- function(N = 100, mean.p = 0.5, T = 3, time.eff = runif(T, -2,
  2)) {
  yfull <- yobs <- array(NA, dim = c(N, T))
  p.vec <- array(NA, dim = T)
```

```

for (j in 1:T) {
  p <- plogis(log(mean.p / (1-mean.p)) + time.eff[j])
  yfull[,j] <- rbinom(n=N, size=1, prob=p)
  p.vec[j] <- p
}
ever.detected <- apply(yfull, 1, max)
C <- sum(ever.detected)
yobs <- yfull[ever.detected == 1,]
cat(C, "out of", N, "animals present were detected.\n")
return(list(N=N, p.vec=p.vec, C=C, T=T, yfull=yfull, yobs=yobs))
}

```

We create one data set. Again, note that each time you do this you will get a slightly different realization of the same stochastic process. It may even (rarely) happen to have all  $N = 100$  individuals captured.

```
data <- data.fn()
```

We do the same kind of analysis as before but just have to add the time effects in the BUGS code.

```

# Augment data set
nz <- 150
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T)))

# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
for (i in 1:T){
  p[i] ~ dunif(0, 1)
}

# Likelihood
for (i in 1:M) {
  z[i] ~ dbern(omega)
  for (j in 1:T){
    yaug[i,j] ~ dbern(p.eff[i,j])
    p.eff[i,j] <- z[i] * p[j]
    } #j
} #i

# Derived quantities
N <- sum(z[])
}

", fill = TRUE)
sink()

# Bundle data
win.data <- list(yaug = yaug, M = nrow(yaug), T = ncol(yaug))

```

```

# Initial values
inits<- function() list(z = rep(1, nrow(yaug)), p = runif(data$T, 0, 1))

# Parameters monitored
params <- c("N", "p", "omega")

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
            n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
            bugs.dir, working.directory=getwd())

# Summarize posteriors
print(out, dig=3)
hist(out$sims.list$N, nclass=40, col="gray", main="", xlab =
    "Population size", las=1, xlim=c(70, 150))
abline(v=data$C, col="black", lwd=3)
[...]
      mean      sd     2.5%    25%    50%    75%   97.5%   Rhat n.eff
N    99.854  9.321  85.000  93.000  99.000 105.000 122.000 1.002 1500
p[1]  0.454  0.063  0.333  0.411  0.453  0.497  0.580 1.002 1900
p[2]  0.346  0.055  0.246  0.307  0.343  0.382  0.464 1.001 3000
p[3]  0.326  0.055  0.224  0.288  0.325  0.362  0.439 1.001 3000
omega 0.445  0.052  0.354  0.408  0.441  0.474  0.565 1.001 2000
[...]

```

So, all is similar as before. Note that an important contribution to the uncertainty about population size  $N$  comes from the uncertainty about detection probability  $p$ . The more we need to estimate for  $p$ , that is, the more parameters we need to describe the patterns in  $p$ , the less precise will be our estimate of  $N$ . You can fit the model  $M_0$  to this same data set to confirm this. Therefore, when designing a study, it is always good to try and eliminate as many factors that will introduce variance in  $p$  as possible. Of course, we can always model such factors, but we pay with reduced precision.

In the above model, we have assumed that the time effects are fixed (although for convenience we simulated detection probabilities from a uniform random number generator). It would be straightforward to fit random instead of fixed time effects: just add a common prior distribution to the set of detection probabilities and then estimate the hyperparameters of that prior distribution. It is customary to model detection probabilities on a transformed scale and assume a normal distribution for the random time effects. The usual transformation is the logit, that is,  $\log[p/(1-p)]$ . We leave this for one of the exercises for you to try out; see also Section 7.4.2.

### 6.2.3 Behavioral or Memory Effects: Model M<sub>b</sub>

Another possible effect is within-capture-history dependence of detection probability  $p$ . This model is called model M<sub>b</sub> by Otis et al. (1978). It fits a different parameter for  $p$  depending on whether an animal was caught before or not. We need to distinguish between immediate (or ephemeral) trap response and permanent trap response. We might also envision something in between and make  $p$  a function of the number of times that an animal was caught over the last, say, 3 or 5 or 10 occasions. In this example, we model immediate trap response, that is, when an individual is captured, it has a different detection probability only on the immediately following occasion, but not thereafter, unless it is captured again.

In the following function, we simulate trap response on the probability scale. We could also do this on the logit scale, which would be more natural if we were to combine trap response with other effects. We denote the capture probability as  $c$  or  $p$  depending on whether an individual has or has not been captured during the preceding occasion.

```
# Define function to simulate data under Mb
data.fn <- function(N = 200, T = 5, p = 0.3, c = 0.4) {
  yfull <- yobs <- array(NA, dim = c(N, T))
  p.eff <- array(NA, dim = N)

  # First capture occasion
  yfull[, 1] <- rbinom(n = N, size = 1, prob = p)

  # Later capture occasions
  for (j in 2:T) {
    p.eff <- (1 - yfull[, (j-1)]) * p + yfull[, (j-1)] * c
    yfull[, j] <- rbinom(n = N, size = 1, prob = p.eff)
  }

  ever.detected <- apply(yfull, 1, max)
  C <- sum(ever.detected)
  yobs <- yfull[ever.detected == 1,]
  cat(C, "out of ", N, "animals present were detected.\n")
  return(list(N = N, p = p, c = c, C = C, T = T, yfull = yfull, yobs = yobs))
}
```

We create one data set with trap-happiness ( $p < c$ ). You may want to look at the data set created and try to see how trap response induces serial autocorrelation in the individual detection histories. This needs to be accounted for in the analysis.

```
data <- data.fn(N = 200)

# Augment data set
nz <- 150
yaug <- rbind(data$yobs, array(0, dim = c(nz, data$T)))

# Specify model in BUGS language
sink("model.txt")
cat("
model {
```

```

# Priors
omega ~ dunif(0, 1)
p ~ dunif(0, 1)      # Cap prob when not caught at t-1
c ~ dunif(0, 1)      # Cap prob when caught at t-1

# Likelihood
for (i in 1:M) {
  z[i] ~ dbern(omega)

  # First occasion
  yaug[i,1] ~ dbern(p.eff[i,1])
  p.eff[i,1] <- z[i] * p

  # All subsequent occasions
  for (j in 2:T) {
    yaug[i,j] ~ dbern(p.eff[i,j])
    p.eff[i,j] <- z[i] * ( (1-yaug[i,(j-1)]) * p + yaug[i,(j-1)] * c )
  } #j
} #i

# Derived quantities
N <- sum(z[])
trap.response <- c - p
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(yaug = yaug, M = nrow(yaug), T = ncol(yaug))

# Initial values
inits <- function() list(z = rep(1, nrow(yaug)), p = runif(1, 0, 1))

# Parameters monitored
params <- c("N", "p", "c", "trap.response", "omega")

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
  bugs.dir, working.directory=getwd())

# Summarize posteriors
print(out, dig=3)
[...]
      mean     sd    2.5%    25%    50%    75%   97.5%   Rhat n.eff
N       203.129 11.878 183.000 195.000 202.000 210.000 230.000 1.010  230
p        0.273  0.026   0.223   0.255   0.273   0.291   0.325 1.008  280
c        0.416  0.032   0.356   0.395   0.417   0.438   0.477 1.002 1300
trap.response 0.143  0.041   0.065   0.115   0.144   0.170   0.225 1.003  830
omega    0.652  0.046   0.567   0.620   0.650   0.682   0.745 1.007  350
[...]

```

```
hist(out$sims.list$N, nclass = 40, col = "gray", main = "", xlab =
  "Population size", las = 1, xlim = c(150, 300))
abline(v = data$C, col = "black", lwd = 3)
```

### 6.2.4 Individual (Random) Effects: The Heterogeneity Model $M_h$

The third model in the Otis et al. (1978) catalog is often called the “heterogeneity model”, or  $M_h$  for short. This term is misleading, since it suggests a single model. However, a multitude of potential models can all be subsumed under the term  $M_h$ . What  $M_h$  means is that we assume that each individual has its own detection probability, and that this heterogeneity cannot be described by known and measured covariates; instead, we assume that there are individual latent (random) effects in detection probability.

There are various possible statistical descriptions of this “diffuse” heterogeneity, for instance, finite mixtures (Pledger, 2000) and beta binomial or logistic-normal continuous mixtures (Coull and Agresti, 1999; Dorazio and Royle, 2003). However, Link (2003) has shown that population size  $N$  is unfortunately not an identifiable parameter *across different classes of models* for individual heterogeneity in  $p$ , such as finite mixtures, beta binomial, or logistic-normal continuous mixtures. This means that models with a different specification of individual heterogeneity may well give very different answers about  $N$ , and yet, we have no data-based criterion to choose among them, such as likelihood ratio tests or AIC. However, we believe that simply ignoring individual heterogeneity would mean to throw out the baby with the bathtub, since not specifying effect  $h$  would lead to underestimated population size (Williams et al., 2002) and thus probably to worse inference, on average, than specifying the “wrong” mixture distribution (Kéry, 2011a).

Here, we consider one such mixture model, the logistic-normal (Coull and Agresti, 1999), which is a continuous mixture model that allows flexible modeling of individual effects along with others, such as time or behavior effects (see [Section 6.3](#)). In this model, individual heterogeneity is modeled as random noise around some mean on a logit-transformed scale, and the model for the noise is the normal distribution. Therefore, it is called logistic normal. OpenBUGS contains another example of model  $M_h$  with data augmentation ([Examples > Ecology examples > Birds](#)).

In our example, we aggregate the capture-histories to capture frequencies, that is, counts of the number of times each individual was encountered. In the absence of factors that induce time- or trap-dependent effects on detection, we do not need to model the binary detection histories and can parsimoniously model capture frequencies. In our code, `mean.p` is the average detection probability of individuals in the population studied, and `sd` is

the standard deviation of the normal distribution which describes the heterogeneity in the individual detection probability on the logit scale. That is, we will estimate a normal distribution for individual detection probability with mean equal to `logit(mean.p)` and standard deviation equal to `sd`.

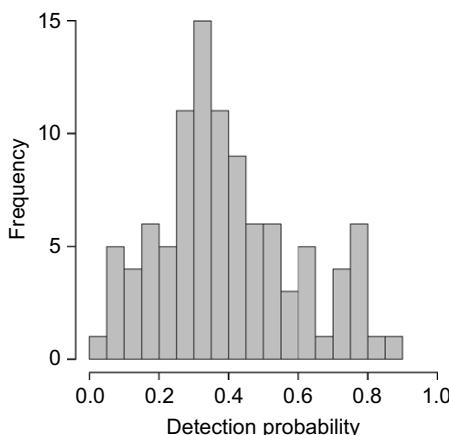
```
# Define function to simulate data under Mh
data.fn <- function(N = 100, mean.p = 0.4, T = 5, sd = 1) {
  yfull <- yobs <- array(NA, dim = c(N, T) )
  mean.lp <- log(mean.p / (1 - mean.p))
  p.vec <- plogis(mean.lp + rnorm(N, 0, sd))

  for (i in 1:N){
    yfull[i,] <- rbinom(n = T, size = 1, prob = p.vec[i])
  }

  ever.detected <- apply(yfull, 1, max)
  C <- sum(ever.detected)
  yobs <- yfull[ever.detected == 1,]
  cat(C, "out of ", N, "animals present were detected.\n")
  hist(p.vec, xlim = c(0,1), nclass = 20, col = "gray", main = "", xlab =
    "Detection probability", las = 1)
  return(list(N = N, p.vec = p.vec, mean.lp = mean.lp, C = C, T = T, yfull =
    yfull, yobs = yobs))
}
```

We create one data set and get a summary of how many individuals were ever detected and a histogram of the individual detection probabilities (Fig. 6.3).

```
data <- data.fn()
84 out of 100 animals present were detected.
```



**FIGURE 6.3** Distribution of individual detection probability for 100 simulated individuals.

We aggregate the capture-histories to capture frequencies and sort them for convenience. Then, we run WinBUGS on the model. As usual, random-effects models require longer Markov chains to achieve convergence. They also require more data augmentation: the posterior for  $N$  is more drawn out (there is more uncertainty about  $N$ ) and so to avoid truncation of the posterior, we have to allow for more potential individuals by increasing the size of  $M$ .

```

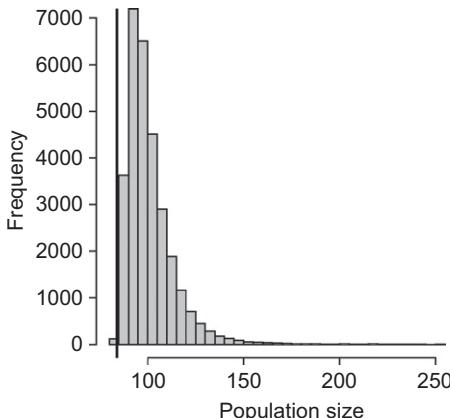
# MCMC settings
ni <- 25000
nt <- 2
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 6 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
            n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
            bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out, dig=3)
hist(out$sims.list$N, nclass=50, col="gray", main="", xlab =
    "Population size", las=1, xlim=c(80, 250))
abline(v=data$C, col="black", lwd=3)
[...]
      mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
N     101.584 12.927 87.000 93.000 98.000 106.000 134.000 1.015 590
mean.p 0.405  0.074  0.225  0.364  0.416  0.457  0.517 1.023 830
sd     1.091  0.355  0.510  0.845  1.050  1.289  1.904 1.017 260
omega   0.266  0.040  0.205  0.239  0.261  0.286  0.358 1.007 950
[...]

```

We see that the precision for  $N$  is lower compared with that under the previous models. Estimation in individual-effects models is notoriously more challenging than in models without such random effects. We see this in longer run time to get an adequate posterior sample as well as in the drawn-out posterior distribution of the key parameter  $N$  (Fig. 6.4).



**FIGURE 6.4** Posterior distribution of population size  $N$  under the heterogeneity model  $M_h$  (black line: observed number of individuals). The posterior mass does not extend below the observed number of individuals (84).

### 6.2.5 Combined Effects: Model $M_{th}$

Time, behavioral, and individual effects may also be combined pairwise or all three, giving rise to what Otis et al. (1978) have called models  $M_{th}$  and so forth. Again, this terminology is useful, but may also be misleading, since it may appear as if there was a single such model. Instead, all that a term such as  $M_{th}$  means is that a model includes the effects of both time and individual heterogeneity. This term does not tell us about how time and individual heterogeneity is parameterized, for instance, whether time effects are fixed or random or whether a finite or a continuous mixture is assumed for individual latent effects. It also does not specify whether the two effects are assumed to be independent or to act in a combined way (i.e., whether time and heterogeneity act as main effects only or with an interaction).

Here, we give one example of a model  $M_{th}$ , which may be more precisely described as having additive (i.e., main) fixed effects of time and random (logistic normal) effects of individuals. This is a useful model in many circumstances in the absence of behavioral effects. Now, we consider effects along both dimensions of the capture-history matrix (corresponding to Table 6.1, with columns representing time, and rows representing individuals). so, we model the unaggregated data. Furthermore, we will model all effects on a logistic scale.

```
# Define function to simulate data under Mth
data.fn <- function(N = 100, T = 5, mean.p = 0.4, time.effects = runif(T,
  -1, 1), sd = 1) {
  yfull <- yobs <- array(NA, dim = c(N, T))
  mean.lp <- log(mean.p / (1 - mean.p))           # mean p on logit scale
  eps <- rnorm(N, 0, sd)                            # Individual effects
  for (j in 1:T) {
    pp <- p[, j] <- plogis(mean.lp + time.effects[j] + eps)
    yfull[, j] <- rbinom(n = N, size = 1, prob = pp)
  }
  ever.detected <- apply(yfull, 1, max)
  C <- sum(ever.detected)
  yobs <- yfull[ever.detected == 1,]
  cat(C, "out of", N, "animals present were detected.\n")
  cat("Mean p per occasion:", round(apply(p, 2, mean), 2), "\n")
  par(mfrow = c(2, 1))
  plot(plogis(mean.lp + time.effects), xlab = "Occasion", type = "b",
    main = "Approx. mean p at each occasion", ylim = c(0, 1))
  hist(plogis(mean.lp + eps), xlim = c(0, 1), col = "gray", main =
    "Approx. distribution of p at average occasion")
  return(list(N = N, mean.lp = mean.lp, time.effects = time.effects,
    sd = sd, eps = eps, C = C, T = T, yfull = yfull, yobs = yobs))
}
```

As shown below, this function can also be used to generate data sets under models  $M_0$ ,  $M_t$ , and  $M_h$  by writing as one or two arguments

`time.effects = runif(5, 0, 0)` and `sd = 0` (you have to adjust  $T$  manually). Here, we create one data set under the model with both effects present and analyze that. We avoid to use the WinBUGS logit function and instead define the logit explicitly, which may sometimes avoid numerical problems.

```

data <- data.fn()
85 out of 100 animals present were detected.
Mean p per occasion: 0.35 0.41 0.47 0.61 0.45

# data<-data.fn(T=10, mean.p=0.2, time.effects=runif(10, 0, 0),
#                 sd=0) # M0
# data<-data.fn(T=10, mean.p=0.5, time.effects=runif(10, 0, 0),
#                 sd=1) # Mh
# data <- data.fn(T=10, sd=0) # Mt

# Augment data set
nz <- 300
yaug <- rbind(data$yobs, array(0, dim=c(nz, data$T)))

# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
for (j in 1:T){
    mean.lp[j] <- log(mean.p[j] / (1 - mean.p[j])) # Define logit
    mean.p[j] ~ dunif(0, 1)
}
tau <- 1 / (sd * sd)
sd ~ dunif(0,5)

# Likelihood
for (i in 1:M){
    z[i] ~ dbern(omega)
    eps[i] ~ dnorm(0, tau) I(-16, 16) # See web appendix A in Royle (2009)
    for (j in 1:T){
        lp[i,j] <- mean.lp[j] + eps[i]
        p[i,j] <- 1 / (1 + exp(-lp[i,j])) # Define logit
        p.eff[i,j] <- z[i] * p[i,j]
        y[i,j] ~ dbern(p.eff[i,j])
    } #j
} #i

# Derived quantities
N <- sum(z[])
}
", fill = TRUE)
sink()

# Bundle data
win.data <- list(y = yaug, M = nrow(yaug), T = ncol(yaug))

# Initial values
inits <- function() list(z = rep(1, nrow(yaug)), sd = runif(1, 0.1, 0.9))

```

```

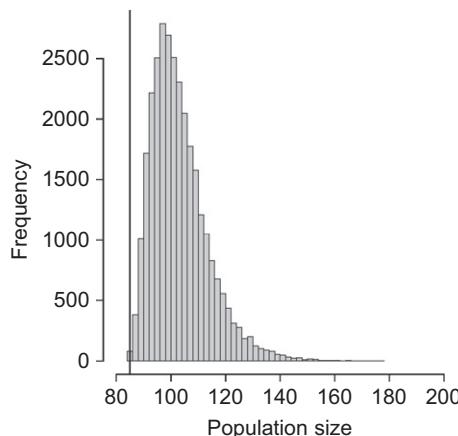
# Parameters monitored
params <- c("N", "mean.p", "mean.lp", "sd", "omega")

# MCMC settings
ni <- 25000
nt <- 2
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 47 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
            n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
            bugs.dir, working.directory=getwd())

# Summarize posteriors
print(out, dig=3)
hist(out$sims.list$N, nclass=50, col="gray", main="", xlab =
    "Population size", las=1, xlim=c(80, 200))
abline(v=data$C, col="black", lwd=3)
[...]
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
N       103.665 10.416 89.000 96.000 102.000 109.000 129.000 1.001 6900
mean.p[1] 0.381 0.083 0.214 0.325 0.383 0.439 0.540 1.001 13000
mean.p[2] 0.419 0.085 0.244 0.363 0.422 0.478 0.578 1.001 30000
mean.p[3] 0.419 0.086 0.243 0.361 0.422 0.479 0.578 1.001 12000
mean.p[4] 0.582 0.089 0.388 0.527 0.590 0.645 0.736 1.001 9000
mean.p[5] 0.322 0.078 0.169 0.269 0.322 0.374 0.474 1.001 21000
mean.lp[1] -0.500 0.371 -1.301 -0.731 -0.476 -0.244 0.162 1.001 10000
mean.lp[2] -0.338 0.365 -1.132 -0.562 -0.314 -0.086 0.313 1.001 20000
mean.lp[3] -0.340 0.369 -1.138 -0.570 -0.314 -0.083 0.315 1.001 9200
mean.lp[4] 0.342 0.374 -0.454 0.109 0.362 0.597 1.026 1.001 6600
mean.lp[5] -0.772 0.376 -1.590 -1.002 -0.746 -0.514 -0.104 1.001 18000
sd        1.411 0.347 0.795 1.173 1.386 1.624 2.173 1.003 2700
omega     0.270 0.035 0.211 0.246 0.267 0.291 0.348 1.001 19000
[...]

```



**FIGURE 6.5** Posterior distribution of population size  $N$  under model  $M_{\text{th}}$  (black line: observed number of individuals).

We obtain decent parameter estimates (see Fig. 6.5 and preceding posterior summary), judging by their resemblance to the parameter values with which we generated our data set.

### **6.3 ANALYSIS OF A REAL DATA SET: MODEL $M_{tbh}$ FOR SPECIES RICHNESS ESTIMATION**

To illustrate estimation of the size of a real population, we will look at the size of a bird community. It has been pointed out repeatedly that the number of species in a community is analogous to the number of individuals in a population (e.g., Boulinier et al., 1998; Nichols et al., 1998a). Therefore, the same inferential framework, closed-population models (though not, for instance, distance sampling), can be used for both (Kéry, 2011a). One important consideration specific to species richness estimation is that a model that does not allow for individual heterogeneity will probably not be adequate: species in a community usually differ by many factors that determine their detection probability, most of all perhaps in their abundance. Thus, detection probability is likely to vary a great deal more among the species in a community than among the individuals in a population. Since not allowing for heterogeneity in  $p$  yields severe underestimates of  $N$ , species richness estimation with closed capture–recapture models should always be based on a heterogeneity model.

We will use bird point count data from the Czech republic (data courtesy of Jiri Reif). With some colleagues, Jiri cycled an East–West transect spanning almost the whole country and conducted a 5 min point count after every 500 m in 2004–2005 (768 points in total). They repeated this five times within the same breeding season. A total of 146 species were detected (Reif et al., 2008), among them the wryneck (Fig. 6.6). We will use their data from one such point: point count number 610. For each species and occasion, the data contain the number of individuals counted. For the analysis, we first reduce these data to simple detection/nondetection data.

```
# Read in data and look at them
p610 <- read.table("p610.txt", header = TRUE)
y <- p610[,5:9]                                # Grab counts
y[y > 1] <- 1                                    # Counts to det-nondetections
C <- sum(apply(y, 1, max)) ; print(C)           # Number of observed species
table(apply(y, 1, sum))                          # Capture-frequencies
```

The data contain the detection histories not only of the 31 species detected at this particular point but also those of all species detected anywhere along the full national transect. Hence, this data set is “naturally data augmented”. We will not add any more zeroes because 115 zeroes corresponding to the species not detected at point 610, but detected somewhere else in the Czech



**FIGURE 6.6** Wryneck (*Jynx torquilla*), Latvia, 2004 (Photograph by T. Muukonen).

Republic, are probably enough. But we can easily check that by inspecting the posterior distribution for  $N$ .

All models in this chapter assume closure (here: community closure). In the present context, this means that each species that is part of the sampled community at a given point must be available for detection during all replicate surveys. This may well not be true. For instance, many bird species are migrants, and some may not yet have arrived during early surveys. If typical arrival dates of each migrant species are known, then this problem could be dealt with by simply turning the corresponding data into missing values. Our data set would then no longer be balanced, but this in general poses no problem to the estimation framework, as long as some replicate surveys are available for some species at least.

Here, we will ignore this possibility and assume closure. We adopt a model  $M_{tbh}$  which has additive effects of time, behavior, and individual heterogeneity on the detection probability of a species. The test for behavioral effects is particularly interesting, since it has been suggested that in monitoring programs, people may detect species detected previously more or less easily thereafter (Riddle et al., 2010).

```
# Specify model in BUGS language
sink("M_tbh.txt")
cat("
model {
```

```

# Priors
omega ~ dunif(0, 1)
for (j in 1:T) {
  alpha[j] <- log(mean.p[j] / (1-mean.p[j])) # Define logit
  mean.p[j] ~ dunif(0, 1) # Detection intercepts
}
gamma ~ dnorm(0, 0.01)
tau <- 1 / (sd * sd)
sd ~ dunif(0, 3)

# Likelihood
for (i in 1:M) {
  z[i] ~ dbern(omega)
  eps[i] ~ dnorm(0, tau) I(-16, 16)

  # First occasion: no term for recapture (gamma)
  y[i,1] ~ dbern(p.eff[i,1])
  p.eff[i,1] <- z[i] * p[i,1]
  p[i,1] <- 1 / (1 + exp(-lp[i,1]))
  lp[i,1] <- alpha[1] + eps[i]

  # All subsequent occasions: includes recapture term (gamma)
  for (j in 2:T) {
    y[i,j] ~ dbern(p.eff[i,j])
    p.eff[i,j] <- z[i] * p[i,j]
    p[i,j] <- 1 / (1 + exp(-lp[i,j]))
    lp[i,j] <- alpha[j] + eps[i] + gamma * y[i,(j-1)]
  } #j
} #i

# Derived quantities
N <- sum(z[])
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(y = as.matrix(y), M = nrow(y), T = ncol(y))

# Initial values
inits <- function() list(z = rep(1, nrow(y)), sd = runif(1, 0.1, 0.9))

# Parameters monitored
params <- c("N", "mean.p", "gamma", "sd", "omega")

# MCMC settings
ni <- 50000
nt <- 4
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 24 min)
out <- bugs(win.data, inits, params, "M_tbh.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

```

```
# Summarize posteriors and plot posterior for N
print(out, dig = 3)
par(mfrow = c(1, 2))
hist(out$sims.list$N, breaks = 35, col = "gray", main = "", xlab =
    "Community size", las = 1, xlim = c(30, 100), freq = FALSE)
abline(v = C, col = "black", lwd = 3)
[...]
      mean     sd   2.5%   25%   50%   75%  97.5% Rhat n.eff
N       42.050 7.444 33.000 37.000 40.000 45.000 61.000 1.002 2300
mean.p[1] 0.241 0.084 0.096 0.181 0.235 0.294 0.420 1.001 13000
mean.p[2] 0.294 0.096 0.126 0.224 0.288 0.356 0.495 1.002 3100
mean.p[3] 0.295 0.097 0.125 0.225 0.289 0.358 0.502 1.002 2200
mean.p[4] 0.222 0.084 0.083 0.161 0.214 0.274 0.408 1.002 1600
mean.p[5] 0.319 0.099 0.140 0.248 0.315 0.385 0.522 1.002 3200
gamma    -0.080 0.496 -1.062 -0.409 -0.099 0.268 0.892 1.010 300
sd       0.709 0.420 0.044 0.387 0.676 0.983 1.622 1.005 11000
omega    0.291 0.062 0.193 0.248 0.283 0.325 0.435 1.002 3100
[...]
```

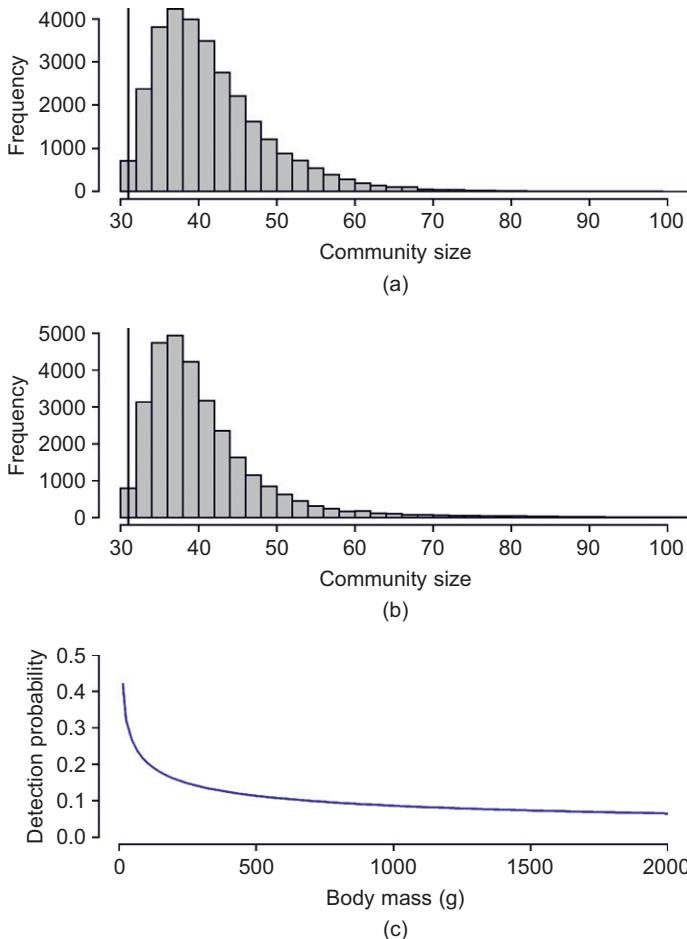
After 24 min worth of sampling the joint posterior distribution of parameters, estimates look decent, and so we make a first interpretation of these results. With 31 species detected, we estimate there were in fact 42, therefore, only  $31/42 = 74\%$  were ever detected (Fig. 6.7a). Given the short survey duration, this appears reasonable. The average detection probability of a species in the community was between 0.22 and 0.32 and did not appear to vary much (in view of the wide uncertainty around these estimates). No behavioral (“memory”) effect on detection probability was discernible (95% CRI of gamma: -1.06 to 0.89). Finally, the standard deviation of the heterogeneity distribution was estimated at 0.71, which, not surprisingly, is considerable; species differ greatly.

It appears likely that whether or not “memory” exists should partly be related to scale. If a single person goes out to a small area and surveys on consecutive days then maybe he remembers what he saw. But, over large scales, over long time, and sampling many species, it is probably less important (Royle, pers. comm.). Since behavioral effects appeared to be absent, we might simplify the model and fit a time and heterogeneity model (i.e.,  $M_{th}$ ).

We have mentioned that inference under model  $M_0$  will often lead to a serious negative bias in the estimate of  $N$  when there is heterogeneity among species in  $p$ . We illustrate this next by fitting model  $M_0$ .

```
# Define model
sink("M0.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
p ~ dunif(0, 1)
```



**FIGURE 6.7** Analysis of community size (species richness,  $N$ ) at point 610 in the Czech transect data (courtesy of Jiri Reif). Posterior distribution of  $N$  (a) under a model with time, behavioral, and heterogeneity effects ( $M_{t\text{bh}}$ ) and (b) under a model with effects of time and the individual covariate body mass ( $M_{t+x}$ ). The black line denotes the observed number of species. (c) Effect of body mass (g) on detection probability.

```
# Likelihood
for (i in 1:M) {
  z[i] ~ dbern(omega)
  for (j in 1:T) {
    y[i,j] ~ dbern(p.eff[i,j])
    p.eff[i,j] <- z[i] * p
  } #j
} #i
```

```

# Derived quantities
N <- sum(z[])
} # end model
",fill = TRUE)
sink()

# Initial values
inits <- function() list(z = rep(1, nrow(y)))

# Define parameters to be monitored
params <- c("N", "p", "omega")

# MCMC settings
ni <- 50000
nt <- 4
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
out0 <- bugs(win.data, inits, params, "M0.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = FALSE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Inspect output
print(out0, dig = 3)
[...]
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
N    37.877 3.970 32.000 35.000 37.00 40.000 47.000 1.001 21000
p     0.301 0.044  0.216  0.271  0.30  0.331  0.390 1.001 16000
omega 0.263 0.045  0.182  0.231  0.26  0.291  0.358 1.001 30000
[...]

```

Under model  $M_0$ , we estimate 38 instead of 42 species. This illustrates the point that ignoring individual heterogeneity in detection probability produces underestimates of, and too short standard errors for, population size  $N$ .

## 6.4 CAPTURE–RECAPTURE MODELS WITH INDIVIDUAL COVARIATES: MODEL $M_{t+x}$

The old classification of Otis et al. (1978) of closed-population models does not include individual covariates. However, continuous detection covariates may often be available, and their effects could be modeled. The challenge is that the covariate values for undetected individuals are not known. Next, we illustrate a model described by Royle (2009), who uses data augmentation to analyze the joint likelihood of the capture-histories and the covariate values. In this model, we describe the distribution of the covariate values in the statistical population in addition to the probabilistic description of the capture-histories. We will give two illustrations for this model: first, in the context of species

richness estimation for the Czech transect data and second, in the context of population size estimation in a mussel.

### 6.4.1 Individual Covariate Model for Species Richness Estimation

In our first example, we model the effect of body mass on detection probability in the Czech bird data. Body mass could affect detection probability  $p$  in various ways. First, everything else equal, larger birds are rarer than smaller birds; hence, we would expect  $p$  to be lower than for smaller birds. Second, larger birds have larger territories, so their  $p$  will contain an important availability component (Kéry and Schmidt, 2008): when they are in a part of their territory that is not sampled, they cannot be detected, and this will lower their  $p$ . In contrast, and third, a larger bird may intrinsically be more visible than a smaller bird. Thus, we might expect an effect of body size on  $p$ , though its direction may not be obvious.

In the Czech bird example, the total list of species that could occur at point 610 is assumed to be known; it is those 146 species that were ever detected anywhere in the Czech transect study (Reif et al., 2008). Hence, the covariate values for all “individuals” are known, and we could proceed with the modeling without assuming a prior distribution for the covariate. However, more typically, when estimating population size, the covariate values are *not* known for individuals not encountered and hence, to estimate these latent covariate values, we must assume a prior distribution for them and estimate the hyperparameters of the latter. To mimick this situation, we discard all body mass data of the  $146 - 31 = 115$  species that were not detected at point 610.

```
p610 <- read.table("p610.txt", header = TRUE)
y <- p610[, 5:9]                                # Grab counts
y[y > 1] <- 1                                    # Convert to det-nondetections
ever.observed <- apply(y, 1, max)
wt <- p610$bm[ever.observed == 1]                 # Body mass
yy <- as.matrix(y[ever.observed == 1, ]) # Detection histories
dimnames(yy) <- NULL
```

To determine a distribution suitable to describe the species-specific body mass, we cheat a little and inspect the data for all 146 species. Body mass is expected to be proportional to the cube of length, and length measurements are often approximately normally distributed. Therefore, we take the cubic root of body mass. In addition, we take the log. It is obvious that the lognormal is not a fantastic description of the cubic root of body mass, but here, we assume it is adequate, at least for our illustrative purposes.

```
mlog <- mean(log(p610$bm^(1/3)))
sdlog <- sd(log(p610$bm^(1/3)))
hist(p610$bm^(1/3), xlim=c(0, 30), nclass=25, freq=FALSE,
     col="gray")
lines(density(rlnorm(n=10^6, meanlog=mlog, sdlog=sdlog)),
      col="blue", lwd=3)
```

Since we discarded the data on undetected species, we now need to actively augment the data set and add 150 potential species. This yields a detection data set with  $M = 181$  rows. We also need to augment the individual covariate with NAs for individuals 32–181.

```
# Augment both data sets
nz = 150
yaug <- rbind(yy, array(0, dim=c(nz, ncol(yy))))
logwt3 <- c(log(wt^(1/3)), rep(NA, nz))
```

It should in theory be possible to fit the same model as in Section 6.3 with the body mass covariate added. However, trying to squeeze out so many parameter estimates from so little data (31 species) may be asking for too much, and indeed, we failed at getting convergence for this model. Hence, we fit a simpler model with effects of time and the covariate, thus dropping the individual and trap response effects. We center the covariate to facilitate interpretation of the parameters and improve convergence. We also provide part of the prior definition in the BUGS model as data (`prior.sd.upper`), which makes the code more flexible.

```
# Specify model in BUGS language
sink("M_t+X.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
for (j in 1:T){
    alpha[j] <- log(mean.p[j] / (1-mean.p[j]))
    mean.p[j] ~ dunif(0, 1)
}
beta ~ dnorm(0, 0.01)
mu.size ~ dnorm(0, 0.01)
tau.size <- 1 / pow(sd.size, 2)
sd.size ~ dunif(0, prior.sd.upper)      # Provide upper bound as data

# Likelihood
for (i in 1:M){      # Loop over individuals
    z[i] ~ dbern(omega)
    size[i] ~ dnorm(mu.size, tau.size) I(-6, 6)
    for (j in 1:T){ # Loop over occasions
        y[i,j] ~ dbern(p.eff[i,j])
        p.eff[i,j] <- z[i] * p[i,j]
```

```

    p[i,j] <- 1 / (1 + exp(-lp[i,j]))
    lp[i,j] <- alpha[j] + beta * size[i]
  } #j
} #i

# Derived quantities
N <- sum(z[])
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(y=yaug, size=logwt3 - mean(logwt3, na.rm=TRUE), M=
  nrow(yaug), T=ncol(yaug), prior.sd.upper=3)

# Initial values
inits <- function() list(z=rep(1, nrow(yaug)), beta=runif(1, 0, 1),
  mu.size=rnorm(1, 0, 1))

# Parameters monitored
params <- c("N", "mean.p", "beta", "omega", "mu.size", "sd.size")

# MCMC settings
ni <- 50000
nt <- 4
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 19 min)
outX<- bugs(win.data, inits, params, "M_t+X.txt", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
  bugs.dir, working.directory=getwd())

# Summarize posteriors and plot posterior for N
print(outX, dig=3)
      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
N       41.638 10.384 32.000 36.000 39.000 44.000 68.000 1.099   65
mean.p[1] 0.270  0.075  0.138  0.216  0.264  0.318  0.430 1.003 1100
mean.p[2] 0.320  0.080  0.176  0.263  0.316  0.373  0.488 1.003 1200
mean.p[3] 0.321  0.080  0.175  0.263  0.317  0.374  0.487 1.003 1100
mean.p[4] 0.244  0.072  0.120  0.192  0.239  0.290  0.399 1.003 1200
mean.p[5] 0.345  0.083  0.197  0.287  0.341  0.400  0.518 1.003 1000
beta      -1.313  0.875 -3.143 -1.873 -1.308 -0.725  0.346 1.061   40
omega     0.233  0.064  0.152  0.195  0.222  0.256  0.387 1.049 100
mu.size   0.070  0.111 -0.092  0.002  0.055  0.116  0.342 1.083  63
sd.size   0.366  0.065  0.272  0.323  0.356  0.398  0.520 1.019 230
[...]

hist(outX$sims.list$N, breaks=100, col="gray", main="", xlab=
  "Community size", las=1, xlim=c(30, 100), freq=FALSE)
abline(v=31, col="black", lwd=3)

```

We get a very similar estimate of species richness ( $N$ ) as under model  $M_{t\text{bh}}$ , though the uncertainty is now greater (Fig. 6.7b). This might suggests that much of the heterogeneity among species in  $p$  may be explained by body size and its correlates. What about the relationship between  $p$  and

body mass? The latter varies about 5–10,500 g, and we produce predictions of  $p$  in relation to mass (up to 2000 g), averaging over time effects. We find a strong negative effect of body mass on detection probability (Fig. 6.7c).

```

pred.wt <- seq(5, 2000, length.out = 100) # Cov. vals for prediction
pred.wt.st <- log(pred.wt^(1/3)) - mean(logwt3, na.rm = TRUE)
# Transform them in the same was as in the analysis
pred.p <- plogis(log(mean(outX$mean$mean.p) /
  (1 - mean(outX$mean$mean.p))) + outX$mean$beta * pred.wt.st)
# Compute predicted response
plot(pred.wt, pred.p, type = "l", lwd = 3, col = "blue", las = 1,
  frame.plot = FALSE, ylim = c(0, 0.5))

```

The individual covariate model of Royle (2009) is an interesting and flexible model. As usual, the motivation for the introduction of covariates may be to eliminate unexplained heterogeneity or to explore the covariate relationships. Casting the closed-population model as an occupancy model with or without partly unobserved covariate values gives us much flexibility for either. Of course, the distributional assumption for the covariate is very much part of the model. The dependence of the inference upon this assumption should be tested, as did Royle (2009).

#### 6.4.2 Individual Covariate Model for Population Size Estimation

In our second example, we analyze data from a population study of the pen shell (Fig. 6.8), conducted by Iris Hendriks and her colleagues in the Balearic islands in 2007 and 2010 (Hendriks et al., in preparation). The pen shell is a large bivalve living in *Posidonia* meadows in the Mediterranean. Its habitat is increasingly affected by anchors of leisure boats. Hendriks et al. had two teams of divers who each conducted an independent survey of a number of transects and recorded and measured the size (shell width in cm) of each *Pinna* individual encountered. The resulting data consisted of the detection history of each shell (e.g., 1 0, for a shell detected by the first team and missed by the second) along with its width. We expected a positive effect of mussel size on detection probability. We ignored several other potential covariates, such as site and vegetation density, and restricted our analysis to the 143 shells encountered in 2010.

```

# Read in data and look at shell width distribution
pinna <- read.table("pinna.txt", header = TRUE)
y <- cbind(pinna$d1, pinna$d2)
size <- pinna$width
hist(size, col = "gray", nclass = 50, xlim = c(0, 30), freq = FALSE)
lines(density(rnorm(10^6, mean = mean(size), sd = sd(size))),
  col = "blue", lwd = 3)

```



**FIGURE 6.8** Pen shell (*Pinna nobilis*), Spain, 2009 (Photograph by I. Hendriks).

We recycle most code from the Czech bird data analysis in the previous section.

```

# Augment both data sets
nz = 150
yaug <- rbind(y, array(0, dim=c(nz, ncol(y))))
size <- c(size, rep(NA, nz))

# Bundle data
win.data <- list(y=yaug, size=size - mean(size, na.rm=TRUE),
                  M=nrow(yaug), T=ncol(yaug), prior.sd.upper=5)

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT 1 min)
outXX <- bugs(win.data, inits, params, "M_t+X.txt", n.chains=nc,
               n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
                 bugs.dir, working.directory=getwd())

# Summarize posteriors
print(outXX, dig=2)

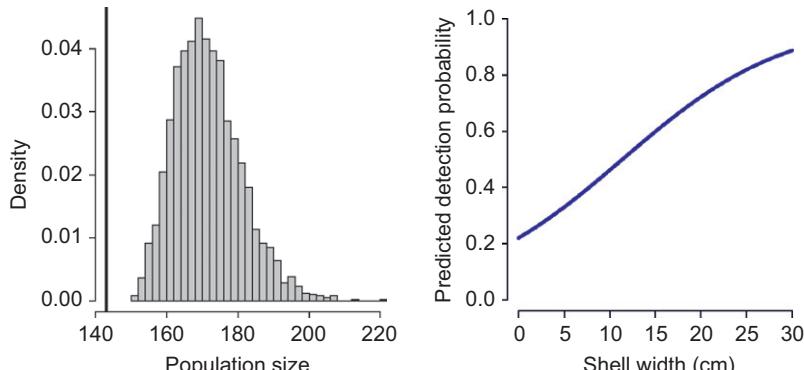
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
N	172.04	9.45	156.00	165.00	171.00	178.00	193.00	1.00	450
mean.p[1]	0.67	0.05	0.57	0.64	0.68	0.71	0.77	1.00	1200
mean.p[2]	0.53	0.05	0.43	0.49	0.53	0.56	0.61	1.00	860
beta	0.11	0.04	0.04	0.09	0.11	0.14	0.18	1.09	50
omega	0.59	0.04	0.50	0.56	0.59	0.61	0.68	1.00	570
mu.size	-0.15	0.26	-0.67	-0.33	-0.16	0.02	0.35	1.01	300
sd.size	3.51	0.15	3.22	3.41	3.51	3.61	3.83	1.00	3000

We estimate that 172 instead of 143 pen shells were available for detection along the surveyed transects. This means that 29 (95% CRI 13–50) were missed by both teams of divers. Detection probability was higher for the first teams, and as expected, there was a positive relationship with shell width (Fig. 6.9).

```
Plot posterior for N and prediction of p
par(mfrow = c(1, 2), mar = c(4.5, 4, 2, 1))
hist(outXX$sims.list$N, breaks = 30, col = "gray", main = "", xlab =
    "Population size", las = 1, xlim = c(143, 220), freq = FALSE)
abline(v = 143, col = "black", lwd = 3)

pred.size <- seq(0, 30, length.out = 1000) # Cov. vals for prediction
pred.size.st <- pred.size - mean(size, na.rm = TRUE) # Transform them
pred.p <- plogis(log(mean(outXX$mean$mean.p) /
    (1 - mean(outXX$mean$mean.p))) + outXX$mean$beta * pred.size.st)
    # Compute predicted detection prob.
plot(pred.size, pred.p, type = "l", lwd = 3, col = "blue", las = 1,
    frame.plot = FALSE, ylim = c(0, 1), xlab = "Shell width (cm)", ylab =
    "Predicted detection probability")
```



**FIGURE 6.9** Analysis of population size (N) of pen shells. Left: Posterior distribution of N (black line: observed number of individuals); right: predicted detection probability in relation to size.

## 6.5 SUMMARY AND OUTLOOK

The population is a central concept in ecology and its size  $N$  perhaps its key descriptor. Because of detection error,  $N$  is usually not directly observable. Capture–recapture methods obtain information about detection probability  $p$  from the repeated encounters of uniquely identifiable individuals and from this derive an estimate of  $N$ . In practice, estimating  $N$  means estimating  $p$  and modeling the key patterns in  $p$ . The same concepts and methods can be used for vastly different kinds of “populations”, including populations of species, that is, communities, where size is equivalent to species richness. In this chapter, we have encountered a catalog of effects (Null, time, behavior, individual heterogeneity) that may be present in  $p$  when estimating  $N$ , including the effects of individual covariates. These concepts are very general, and we will find them in different situations for parameters other than detection probability later in this book. We have also introduced data augmentation (DA) and fitted all models in this chapter using this ingenious but simple technique. DA is an important concept and helps to unify a vast number of capture–recapture type models, which were hitherto regarded as technically distinct procedures. See Royle and Dorazio (2008) to get a flavor of the power of DA.

Obviously, we have only given a very selective view of the large field of population size estimation, and interested readers will have to refer to the many standard references now available. We also briefly note that we do not deal with one important class of models that also achieve a clean accounting of the ecological and the observation processes in estimating density or abundance: distance sampling (Buckland et al., 2001; Williams et al., 2002; Buckland et al., 2004; Royle and Dorazio, 2008). These models are the focus of an active branch of ecological statistics and have much relevance for the analysis of populations. Distance sampling can be implemented in WinBUGS; see Royle and Dorazio (2008). OpenBUGS contains a distance sampling example (Examples > Ecology examples > Impala).

We have assumed that the area with which a population is associated may be delimited unambiguously. This is often not the case and to solve this problem, spatial capture–recapture methods are required, which represent a merging of the capture–recapture class of models with those of the distance sampling-type (Efford, 2004; Borchers and Efford, 2008; Royle and Young, 2008; Gardner et al., 2009; Efford et al., 2009a, 2009b; Royle et al., 2009a, 2009b). These models are very powerful and flexible. One of their advantages is that they can estimate detection probability, and therefore, can estimate abundance from the spatial pattern of multiple detections of individuals alone (Efford et al., 2009b); no temporal replicate surveys are required. This is a huge design advantage! It seems likely that

in the near future we will see much development along this branch of the population assessment tree in ecological statistics. Again, most spatial capture–recapture models can be implemented in WinBUGS in a transparent way. Likelihood inference can be obtained with user-friendly packages such as program Density ([www.otago.ac.nz/density/](http://www.otago.ac.nz/density/)) or the recent R package *secr* developed by Murray Efford.

All models in the current chapter have assumed closed populations, but we may also want to estimate  $N$  in open populations. The Jolly–Seber model (Chapter 10) models population dynamics, that is, survival and recruitment rates and population size, whereas the  $N$ -mixture and site-occupancy models (Chapters 12 and 13) may be used to model  $N$  in collections of open populations.

## 6.6 EXERCISES

1. Rewrite the model  $M_h$  in [Section 6.2.4](#) for encounter history instead of capture frequency data and fit it. (The response will be Bernoulli instead of Binomial.)
2. Try to fit the model with permanent trap response to the bird survey data. Imagine a biological situation that might represent this model, on the part of the observer? On the part of the animal?
3. Check out the behavior of estimators in small sample situations, example, the heterogeneity model with 20 individuals and heterogeneity. Does this work?
4. Generate data with individual heterogeneity in  $p$  and fit model  $M_0$ . See how well  $N$  is estimated.
5. Find out whether a model with trap response and time effects is estimable with  $T = 2$ .
6. And what about pure model  $M_b$  with  $T = 2$ ?
7. In  $M_t$ , adapt both the data generation and the model fitting code to random instead of fixed time effects.
8. Check the effects of assumption violations. Fit a model to a data set that was not generated under the same model. For instance, generate data under model  $M_t$  and analyze the resulting data set under  $M_0$  to see what happens to your estimates of  $N$  and  $p$  when you ignore time variation in  $p$ . Do similar things to other pairs of models.
9. Use the Czech point count data and estimate species richness, where detection is a function of body mass, similar as in [Section 6.4.1](#). But this time, include the body mass of all unobserved species. Hint: you then no longer have to give a prior for body mass. Does the estimate of population size and of detection probability become more precise?

# Estimation of Survival from Capture–Recapture Data Using the Cormack–Jolly–Seber Model

## OUTLINE

<b>7.1</b>	<b>Introduction</b>	<b>172</b>
<b>7.2</b>	<b>The CJS Model as a State-Space Model</b>	<b>175</b>
<b>7.3</b>	<b>Models with Constant Parameters</b>	<b>177</b>
7.3.1	<i>Inclusion of Information about Latent State Variable</i>	181
<b>7.4</b>	<b>Models with Time-Variation</b>	<b>183</b>
7.4.1	<i>Fixed Time Effects</i>	184
7.4.2	<i>Random Time Effects</i>	184
7.4.3	<i>Temporal Covariates</i>	188
<b>7.5</b>	<b>Models with Individual Variation</b>	<b>192</b>
7.5.1	<i>Fixed Group Effects</i>	192
7.5.2	<i>Random Group Effects</i>	194
7.5.3	<i>Individual Random Effects</i>	195
<b>7.6</b>	<b>Models with Time and Group Effects</b>	<b>199</b>
7.6.1	<i>Fixed Group and Time Effects</i>	199
7.6.2	<i>Fixed Group and Random Time Effects</i>	204
<b>7.7</b>	<b>Models with Age Effects</b>	<b>208</b>
<b>7.8</b>	<b>Immediate Trap Response in Recapture Probability</b>	<b>212</b>
<b>7.9</b>	<b>Parameter Identifiability</b>	<b>216</b>

<b>7.10 Fitting the CJS to Data in the M-Array Format: the Multinomial Likelihood</b>	<b>220</b>
7.10.1 Introduction	220
7.10.2 Time-Dependent Models	222
7.10.3 Age-Dependent Models	227
<b>7.11 Analysis of a Real Data Set: Survival of Female Leisler's Bats</b>	<b>231</b>
<b>7.12 Summary and Outlook</b>	<b>237</b>
<b>7.13 Exercises</b>	<b>238</b>

## 7.1 INTRODUCTION

---

The preceding chapters dealt with the modeling and estimation of population size and with the simplest summary of population dynamics: population trend. The following chapters focus on one of the main components of population dynamics: survival probability. Survival probability is a key demographic parameter and can have a strong impact on population dynamics (Clobert and Lebreton, 1991; Saether and Bakke, 2000). Typically, interest focuses on estimation (what is the survival in that population?) as well as on modeling, for example, to test whether survival changes with age or differs between groups of individuals or regions, and to estimate how strongly it varies over time or what proportion of temporal variability can be explained by an external covariate such as weather.

In principle, survival estimation is fairly simple—we just have to count the number of individuals alive at a given time  $t$  ( $C_t$ ), and keep track of how many of them die ( $D_{\Delta t}$ ) during the period  $\Delta t$  for which we wish to estimate survival. Sometimes, it may be easier to count the number of the  $C_t$  that are still alive at  $t + \Delta t$ , that is, the number that survived the period  $\Delta t$  ( $L_{\Delta t}$ ). Then survival probability  $s_t$  is

$$s_t = \frac{C_t - D_{\Delta t}}{C_t} = \frac{L_{\Delta t}}{C_t}$$

These numbers may easily be obtained in humans, but they are difficult to get in animal or plant populations. The reason for that is because the detection of individuals is usually far from perfect, so when an individual is not seen, we don't know whether it is dead or still alive. Therefore, such simple calculations cannot often be used, and we need to account for the

observation process in our inferences about survival (an exception being data on individuals with radio tags, White and Garrott, 1990). From the number of individuals recorded at  $t$  ( $C_t$ ), we typically detect only a fraction of those still alive at time  $t + \Delta t$ , which is  $p^*L_{\Delta t}$ :  $p$  is the recapture or resighting probability (depending on the context or study design), which needs to be estimated in order to obtain unbiased estimates of survival. Estimating  $p$  becomes possible if we extend the recapture study to at least one further time step ( $t + 2\Delta t$ ). We may then have individuals that are known to have survived until  $t + 2\Delta t$ , but which have not been seen at time  $t + \Delta t$ . Intuitively, it is clear that the proportion of these individuals provides information on  $p$ .

The most common statistical method to jointly estimate recapture and survival probabilities in animal and plant populations is a class of open population capture–recapture models to which the Cormack–Jolly–Seber (CJS) model belongs (Cormack, 1964; Jolly, 1965; Seber, 1965). Re-encounters may be obtained by different methods (physical capture, sightings, genetic tracking), but the key is that individuals are identified without error. That is, we only have false negatives, but no false positives. The frequentist analysis of the CJS model is described in detail in Lebreton et al. (1992) and Williams et al. (2002). Descriptions and examples of the Bayesian analysis of the CJS model can be found in an increasing number of articles and books (Brooks et al., 2000a; McCarthy and Masters, 2005; Gimenez et al., 2007; McCarthy, 2007; Zheng et al., 2007; Royle, 2008; Royle and Dorazio, 2008; Schofield et al., 2009; Gimenez et al., 2009a; King et al., 2010).

The CJS model can be fitted using either a multinomial (Lebreton et al., 1992) or a state-space likelihood (Gimenez et al., 2007; Royle, 2008). Because these two likelihoods are just different ways of describing what is essentially the same model, they are based on the same sampling design and the same underlying model assumptions. The sampling design is as follows: A random sample of individuals from the study population is captured, all are marked individually, and released into the population again. This is repeated several times. The length of the time intervals between repeated capture occasions depends on the research question, as well as on the life history and population dynamics of the study organism, and capture should be instantaneous or over a short time period. Some marked individuals will be re-encountered, and thus, we obtain capture–recapture data that can be summarized in individual capture histories.

The CJS model makes a number of assumptions and, as usual, their violation may bias parameter estimators. Some assumptions must be met at the design stage of a study. Tags or other marks must not be lost, otherwise survival is underestimated. If mark loss is suspected, double marking and corresponding model adaptation that account for mark loss are necessary to get unbiased estimates of survival (e.g., Smout et al., 2011).

Ideally, capture should be instantaneous, otherwise the interval between capture occasions may differ among individuals and, consequently, there will be individual heterogeneity in survival. However, simulation studies have suggested that the violation of this last assumption does often not have a strong effect on parameters estimates (Hargrove and Borland, 1994). The CJS model also assumes that the identity of the individuals is always recorded without errors. If this assumption is violated, bias can go in either direction, and there is no means to correct for it. Finally, captured and recaptured individuals are regarded as a random sample from the study population. This sounds easy, but in practice, it can be difficult to achieve. For example, in studies on birds using nest boxes, it is quite typical that adults are only captured after the young have hatched because they are likely to abandon their brood if they are disturbed at an early stage. This results in a sample that is biased toward successfully breeding adults, which may or may not be a random sample from all adults in the population.

Further assumptions of the CJS model cannot be violated or fulfilled by the design of the study, rather they are a consequence of how the model is specified. The basic model assumes that each individual within an age class or group has the same survival and recapture probability. Goodness-of-fit tests help to identify severe violation of these assumptions (e.g., trap-response: Pradel, 1993; transients: Pradel et al., 1997), and modifications to the model allow to account for these violations. Individuals must behave independently from each other in terms of survival and recapture. This may not be the case if members of the same family are included in a sample and especially if they remain in family groups. Violation of this assumption is like pseudo-replication (Hurlbert, 1984). The degree of nonindependence leading to overdispersion can be estimated and the standard errors of the estimates as well as AIC-based model selection can be adjusted accordingly (Anderson et al., 1994).

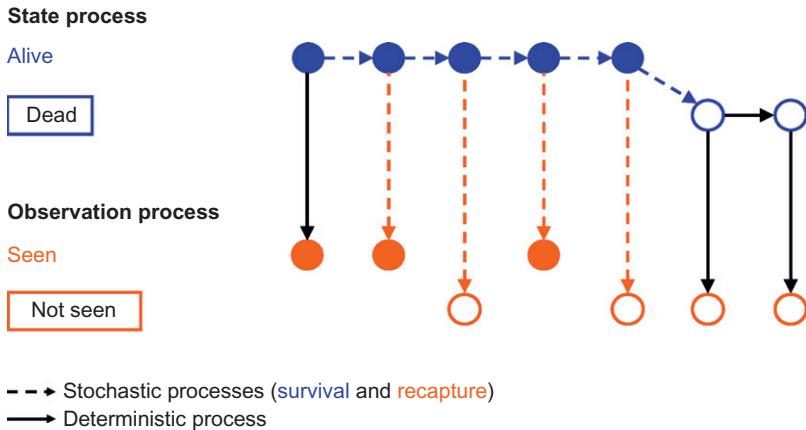
With CJS models, we estimate recapture probability ( $p_t$ ; the probability of catching/resighting a marked individual at  $t$  that is alive and in the sampling population at  $t$ ) and apparent (also called “local”) survival probability ( $\phi_t$ ; the probability that an individual that is alive and in the population at  $t$  is still alive and in the population at time  $t + 1$ ). Mortality and permanent emigration are confounded, and therefore apparent survival is always lower than true survival whenever permanent emigration is not zero. The difference between apparent and true survival is a matter of study design. Generally, the larger a study area the closer the match between apparent and true survival because dispersing individuals have a higher probability to remain in the study area (Marshall et al., 2004). Throughout this chapter, we will often just write “survival” for ease of presentation—but it is important to remember that survival in the CJS model always refers to a study area.

In this chapter, we introduce the CJS model and illustrate how it is fitted using the state-space (Sections 7.2–7.8) and the multinomial likelihood (Sections 7.9–7.11). We highlight advantages and disadvantages of each approach. Moreover, we will repeatedly use the generalized linear (mixed) model (GLM and GLMM) formulations to describe structure in the parameters. This allows the modeling of individual and temporal effects, both of which can either be categorical or continuous, as well as fixed or random. We will also see how correlations among parameters can be modeled through correlated random effects, using a multivariate normal distribution. In most examples, we focus on the modeling of survival, yet, clearly, similar modeling can be conducted for the recapture probability, and all these GLM formulations can also be used in other capture–recapture types of models starting with Chapter 6. Finally, in Section 7.10, we introduce posterior predictive model checking (see Gelman et al., 1996, 2004; Kéry, 2010). This provides a very general framework for the assessment of goodness-of-fit of a model to a data set.

## 7.2 THE CJS MODEL AS A STATE-SPACE MODEL

The state-space formulation of the CJS model has been introduced by Gimenez et al. (2007) and Royle (2008). Let us assume an individual marked at time  $t$ . It may survive until time  $t + 1$  with probability  $\phi_t$ . Conceptually, we can imagine the individual tossing a coin to determine whether it survives (with probability  $\phi_t$ ) or dies (with probability  $1 - \phi_t$ ). Given that the individual is still alive at time  $t + 1$ , it may again survive until  $t + 2$  with probability  $\phi_{t+1}$ . This process is continued until the individual is either dead or the study ends. Clearly, once an individual is dead, its fate is no longer stochastic, and it will remain dead with probability 1. This is the description of the state process, that is, of the states (alive, dead) of an individual over time. We would like to know survival, which requires knowledge of these states of the individuals. Yet, we typically do not have complete information about the true states. A marked individual that is alive at occasion  $t$  may be recaptured (or more generally re-encountered) with probability  $p_t$ . Again, we can imagine that a coin is tossed, determining whether the individual is recaptured (with probability  $p_t$ ) or not (with probability  $1 - p_t$ ). Once an individual is dead, it cannot be recaptured anymore. This is the description of the observation process, which is conditional on the state process, and thus there is a hierarchical structure in the state-space model. In Fig. 7.1, the two processes are shown graphically.

The data observed in a capture–recapture study can be summarized in a capture-history matrix ( $y$ ), which has dimension  $I \times T$ , where  $I$  is the total number of marked individuals, and  $T$  is the number of capture



**FIGURE 7.1** Example of the state and observation process of a marked individual over time for the CJS model. The sequence of true states in this individual is  $z = [1, 1, 1, 1, 1, 0, 0]$ , and the observed capture-history is  $y = [1, 1, 0, 1, 0, 0, 0]$ .

occasions. The matrix entries are either a 1 or a 0. A 1 at position  $i, t$  indicates that individual  $i$  was captured at occasion  $t$ , meaning that it was alive for sure; a 0 at position  $i, t$  shows that individual  $i$  was not captured at  $t$ , meaning that it was either dead, or alive but not caught, or not yet marked.

To estimate survival from such data, we define the latent variable  $z_{i,t}$ , which takes value 1 if individual  $i$  is alive at time  $t$ , and value 0 if it is dead. Thus,  $z_{i,t}$  defines the true state of individual  $i$  at time  $t$ . We also define vector  $f_i$ , which denotes the occasion at which individual  $i$  is first captured (i.e., marked) because only events after first capture are modeled in the CJS model. The state of individual  $i$  at first capture ( $z_{i,f_i}$ ) is 1 with probability 1, as the individual is alive for certain. The states on subsequent occasions are modeled as Bernoulli trials. Conditional on being alive at occasion  $t$ , individual  $i$  may survive until occasion  $t+1$  with probability  $\phi_{i,t}$  ( $t = 1, \dots, T-1$ ). The following two equations define the state process:

$$z_{i,f_i} = 1$$

$$z_{i,t+1} | z_{i,t} \sim \text{Bernoulli}(z_{i,t} \phi_{i,t}).$$

The Bernoulli success parameter is composed of the product of survival and the state variable  $z$ . The inclusion of  $z$  ensures that a dead individual ( $z=0$ ) remains dead and has no further impact on the estimation of survival.

If individual  $i$  is alive at occasion  $t$ , it may be recaptured with probability  $p_{i,t}$  ( $t = 2, \dots, T$ ). This can again be modeled as the realization of a

Bernoulli trial with success probability  $p_{i,t}$ . The following equation defines the observation process:

$$y_{i,t} | z_{i,t} \sim \text{Bernoulli}(z_{i,t} p_{i,t}).$$

The inclusion of the latent variable  $z$  in the Bernoulli trial ensures that dead individuals cannot be encountered. The state and the observation process are both defined for  $t \geq f_i$ . We repeat that the initial capture process is not modeled in the CJS model (see Fig. 7.1) because the initial observation at the time of capture does not contain any information about survival. In contrast, capture-histories at and before initial capture contain information about recruitment, which is a target of estimation of the Jolly–Seber models (see Chapter 10). Because initial capture is not modeled in the CJS model, we say that we condition on first capture.

The implementation of the CJS model in WinBUGS is straightforward. The most general likelihood is based on the above-mentioned three equations and contains different survival and recapture probabilities for each individual at each capture occasion. However, the parameters of this saturated model are not separately estimable (see Section 7.9 for more on this topic), and we need to introduce constraints. These constraints define the structure of the model fitted and may be imposed either along the time or the individual axis of the capture-history matrix, or along both (see Section 6.2). Thus, whatever model we fit, we do not need to change the likelihood, which describes the basic structure of the model, but just these constraints and the corresponding priors. This may not result in code that is the most efficient in terms of computing time and the easiest to read for a beginner. However, with a little practice, it will be seen to be an efficient way of fitting a wide array of models.

### **7.3 MODELS WITH CONSTANT PARAMETERS**

We start with a very simple model, in which survival and recapture, respectively, are identical for all individuals at all occasions. Thus, we impose constraints along both the time and the individual axis of the capture-history matrix. We first simulate the data, and then analyze them. The function to simulate capture–recapture data (`simul.cjs`) is very general and works for all examples in this chapter. We choose the number of individuals released at each occasion. The function then evaluates for each released individual whether it survives, and if so, whether it is recaptured, by two Bernoulli trials governed by individual- and time-specific survival and recapture probabilities that we also provide as input (matrices `PHI` and `P`). Thus, the data-generating function works analogous to the analyzing model.



**FIGURE 7.2** Pair of little owls (*Athene noctua*) (Photograph by H. Sylvain).

In the simulation, we will mimic a study on little owls (Fig. 7.2), a small owl species living in semi-open habitats such as orchards, where it likes to occupy nest boxes. Nest boxes in a study area are checked in May in six study years, and breeding adults are ringed. In each of the six study years, 50 unmarked (new) adults are caught, along with a variable number of individuals that are already marked. Survival of adult little owls is typically around 0.65 (Schaub et al., 2006), and we assume a recapture probability of 0.4. The following R code simulates a matrix with capture-histories. We do not consider individuals first captured on the last occasion, because they do not provide information about survival and recapture.

```
# Define parameter values
n.occasions <- 6                                # Number of capture occasions
marked <- rep(50, n.occasions-1)                  # Annual number of newly marked
                                                       individuals
phi <- rep(0.65, n.occasions-1)
p <- rep(0.4, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked))
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))

# Define function to simulate a capture-history (CH) matrix
simul.cjs <- function(PHI, P, marked) {
  n.occasions <- dim(PHI)[2] + 1
```

```

CH <- matrix(0, ncol = n.occasions, nrow = sum(marked) )
# Define a vector with the occasion of marking
mark.occ <- rep(1:length(marked), marked[1:length(marked)])
# Fill the CH matrix
for (i in 1:sum(marked)){
  CH[i, mark.occ[i]] <- 1           # Write an 1 at the release occasion
  if (mark.occ[i]==n.occasions) next
  for (t in (mark.occ[i]+1):n.occasions){
    # Bernoulli trial: does individual survive occasion?
    sur <- rbinom(1, 1, PHI[i,t-1])
    if (sur==0) break      # If dead, move to next individual
    # Bernoulli trial: is individual recaptured?
    rp <- rbinom(1, 1, P[i,t-1])
    if (rp==1) CH[i,t] <- 1
  } #t
} #i
return(CH)
}

# Execute function
CH <- simul.cjs(PHI, P, marked)

```

Next, we need to create vector  $f$ , which contains the occasion at which each individual is marked.

```

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

```

Finally, we write the BUGS code for a constant model. The two linear models applied are  $\phi_{i,t} = \bar{\phi}$  and  $p_{i,t} = \bar{p}$ . These are in fact the linear predictors (see Chapter 3), but here we call them constraints because we reduce the dimensions of the  $\phi_{i,t}$  and  $p_{i,t}$ , that is, we constrain them. We do not include covariates or random effects, so there is no need for a transformation, and the identity link is applied. The uniform priors ensure that the parameter estimates are in the interval  $[0, 1]$ . The specification of noninformative priors is easy because a uniform ( $U(0, 1)$ ) or a beta distribution ( $\text{beta}(1,1)$ ) can be used. Note that the time indexing in the “Likelihood” part is slightly different to that used in the formulas (see Section 7.2). It avoids the use of separate loops for the state and the observation process.

```

# Specify model in BUGS language
sink("cjs-c-c.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
  for (t in f[i]:(n.occasions-1)){
    phi[i,t] <- mean.phi
  }
}

```

```

    p[i,t] <- mean.p
  } #t
} #i

mean.phi ~ dunif(0, 1)           # Prior for mean survival
mean.p ~ dunif(0, 1)             # Prior for mean recapture

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions) {
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
},fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions =
  dim(CH)[2])

```

Initial values should be given for the two structural parameters and for the latent variable  $z$ . The easiest way for the latter is just to use the observed capture-histories. We have to make sure that initial values for  $z$  are provided only after initial capture. The function below creates the required initial values based on the observed capture-histories and the vector with the occasion of first capture.

```

# Function to create a matrix of initial values for latent state z
ch.init <- function(ch, f) {
  for (i in 1:dim(ch)[1]) {ch[i,1:f[i]] <- NA}
  return(ch)
}

# Initial values
inits <- function(){list(z = ch.init(CH, f), mean.phi = runif(1, 0, 1),
  mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p")

# MCMC settings
ni <- 10000
nt <- 6
nb <- 5000
nc <- 3

```

```
# Call WinBUGS from R (BRT 1 min)
cjs.c.c <- bugs(bugs.data, inits, parameters, "cjs-c-c.bug", n.chains =
  nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())
```

The model does not take a long time to run and convergence is reached after just 5000 iterations. The estimates are very close to the values used for the simulations.

```
# Summarize posteriors
print(cjs.c.c, digits = 3)

  mean     sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
mean.phi 0.679 0.043 0.601 0.649 0.677 0.707 0.767 1.003    980
mean.p   0.370 0.044 0.286 0.340 0.369 0.400 0.458 1.003   1700
```

Sometimes not all individuals recaptured are released again, for instance, when an individual dies at capture. For these individuals, we know that they have survived from initial capture until the last capture; afterward they are not in the sample anymore. This is easy to model and just requires that we define a vector  $h$  which for each individual contains the occasion after which it is not released. For individuals that stay in the sample until the end of the study, the element of vector  $h$  is just the last occasion of the study. Then, vector  $h$  must become an element of the input data and the loop in the likelihood needs to be changed from `for (t in (f[i]+1):n.occasions) { ... }` to `for (t in (f[i]+1):h[i]) { ... }`

### 7.3.1 Inclusion of Information about Latent State Variable

Written as state-space models, CJS models can take a long time to run because there is a loop over all individuals and occasions. The latent state variable  $z$  needs to be updated (estimated) at each MCMC iteration. So far we have treated  $z$  as if we had no information about it. The only information that we included are the observed capture-histories ( $Y$ ), but they are related to  $z$  only through the observation process in the state-space model. Therefore, all elements of  $z$  (i.e., for all individuals after first capture) must be estimated, even when some of them are known.

To improve computation speed and convergence, we can add what we know about the latent state  $z$ , namely, whenever we observe a marked individual we know its latent state is  $z = 1$ . In addition, we know that  $z = 1$  for all occasions between the first and the last observation of an individual, even if it was not seen at all occasions. To include this information in the model (i.e. to prevent estimation of what is not an unknown quantity), we create a matrix that has a value of 1 at all occasions where we know individuals were alive, and NAs elsewhere. The CJS model is conditional on first capture, so the latent state is only defined after first

capture, and thus at all first captures, we need NAs as well. The following function creates the required matrix.

```
# Function to create a matrix with information about known latent state z
known.state.cjs <- function(ch){
  state <- ch
  for (i in 1:dim(ch)[1]){
    n1 <- min(which(ch[i,]==1))
    n2 <- max(which(ch[i,]==1))
    state[i,n1:n2] <- 1
    state[i,n1] <- NA
  }
  state[state==0] <- NA
  return(state)
}
```

This information about  $z$  is then given as data as well.

```
# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
z = known.state.cjs(CH))
```

The initial values for  $z$  now also require some changes: we should not give initial values for those elements of  $z$  whose value is specified in the data; they get an NA.

```
# Function to create a matrix of initial values for latent state z
cjs.init.z <- function(ch,f){
  for (i in 1:dim(ch)[1]){
    if (sum(ch[i,])==1) next
    n2 <- max(which(ch[i,]==1))
    ch[i,f[i]:n2] <- NA
  }
  for (i in 1:dim(ch)[1]){
    ch[i,1:f[i]] <- NA
  }
  return(ch)
}
```

Now, we give initial values for all the quantities to be estimated and run the model:

```
# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p")

# MCMC settings
ni <- 10000
nt <- 6
```

```

nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
cjs.c.c <- bugs(bugs.data, inits, parameters, "cjs-c-c.bug", n.chains =
  nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

# Summarize posteriors
print(cjs.c.c, digits = 3)

  mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
mean.phi 0.675 0.040 0.599 0.648 0.675 0.702 0.754 1.003  2500
mean.p   0.372 0.043 0.294 0.342 0.371 0.399 0.461 1.003  2500

```

The model now runs faster. The difference in run time in this simple case is slight, but time savings can be substantial with more complex models and larger data sets. Therefore, we recommend providing all available information about the latent state in the data. In the following sections of this chapter, we will always, and in most other chapters often, do this. Note that the inclusion of the information about the latent state  $z$  has nothing to do with the use of an informative prior, we simply avoid estimation of known quantities to speed up computation.

## 7.4 MODELS WITH TIME-VARIATION

So far we have fitted the simplest possible model in the CJS family. It assumes that survival and recapture probabilities remain constant over time and are identical for all individuals. In practice, we typically want to relax these strict assumptions. We also may have an interest in fitting models that combine time and individual effects and modeling these effects as additive or interactive. We next consider models with temporal variation, that is, we model the column dimension of the capture-history matrix and constrain the row dimension to be constant (all individuals are treated as identical).

The variation of survival probability from one year to another often has a strong impact on the dynamics of a population. If survival varies much from year to year (i.e., temporal variability is large), population size changes more than when survival probability changes only little over time, all other demographic processes being equal. Thus, there is an interest in measuring temporal variation. Moreover, the annual fluctuations of survival or recapture may be caused by environmental factors that we may have an interest in identifying.

The models to study temporal effects of survival or recapture assume either fixed or random temporal effects, as well as the relationship between focal parameters and temporally varying covariates (e.g., weather).

The fixed-effect time model assumes the parameters to be different at each occasion and independent of each other. This approach is used if there is interest in estimates from particular occasions. By contrast, the model that considers time to be a random effect assumes that time effects are drawn from a statistical distribution, whose parameters we aim to estimate; typically, we will use a normal distribution and estimate a mean and a variance. Therefore, annual estimates are no longer independent from one another. Interest is then not so much in the individual annual effects, but more in an estimation of the mean and the variance of the annual estimates. Fixed- and random-effects models are easily fit within the framework that we have set up (see Chapter 4). The likelihood part of the BUGS code does not need any change at all: all required modifications take place in the “Priors and constraints” section of the BUGS code. In the examples that follow, we will usually model effects on survival only, but of course similar models can be adopted for recapture, too, and any combinations are possible, for example, survival with random time effects and recapture with fixed time effects.

### 7.4.1 Fixed Time Effects

We now assume that survival and recapture vary independently over time, that is, we regard time as a fixed-effects factor. To implement this model, we impose the following constraints:  $\phi_{i,t} = \alpha_t$  and  $p_{i,t} = \beta_t$ , where  $\alpha_t$  and  $\beta_t$  are the time-specific survival and recapture probabilities, respectively. Here is the part of the BUGS model specification that needs to be changed.

```
# Priors and constraints
for (i in 1:nind) {
    for (t in f[i]:(n.occasions-1)) {
        phi[i,t] <- alpha[t]
        p[i,t] <- beta[t]
    } #t
} #i
for (t in 1:n.occasions-1) {
    alpha[t] ~ dunif(0, 1)           # Priors for time-spec. survival
    beta[t] ~ dunif(0, 1)            # Priors for time-spec. recapture
}
```

### 7.4.2 Random Time Effects

The model just shown treats time as a fixed-effects factor; for every occasion, an independent effect is estimated. To assess the temporal variability, we cannot simply take these fixed-effects estimates and calculate their variance. By doing so, we would ignore the fact that these values

are estimates that have an unknown associated error. Thus, we would assume that there is no sampling variance, and this can hardly ever be true (see, e.g., Gould and Nichols, 1998). However, when treating time as a random-effects factor, we can separate sampling (i.e., variance within years) from process variance (i.e., variance between years), exactly as we did in the state-space models in Chapter 5. We model survival or recapture probabilities on the logit scale as a realization of a random process described by a normal distribution with mean  $\mu$  and variance  $\sigma^2$ . The logit link function ensures that the estimated probabilities remain within the interval between 0 and 1:

$$\text{logit}(\phi_{i,t}) = \mu + \varepsilon_t \\ \varepsilon_t \sim \text{Normal}(0, \sigma^2).$$

$\varepsilon_t$  is the deviation from the overall mean survival probability; thus it is a “temporal residual”. The temporal variance ( $\sigma^2$ ) is on the logit scale; thus, it is the temporal variance of the logit survival. Sometimes, one needs an estimate on the probability scale, for instance, when the temporal variance should be compared with the variance of another demographic rate to decide which parameter is more variable over time. A back-transformation is possible by applying the delta method (Powell, 2007). We use

$$\sigma_\theta^2 \cong \sigma^2 \theta^2 (1 - \theta)^2,$$

where  $\theta = \frac{\exp(\mu)}{1 + \exp(\mu)}$  and  $\sigma_\theta^2$  is the variance on the back-transformed scale. It is easy to estimate this quantity directly in BUGS.

To illustrate the approach, we simulate data and analyze them. In the little owl example, we assume a mean survival probability of females of 0.65 and temporal variance of 1 on the logit scale. Reasonable estimates of the temporal variance require a large number of years (>10; Burnham and White, 2002). Here, we simulate data over 20 years.

```
# Define parameter values
n.occasions <- 20 # Number of capture occasions
marked <- rep(30, n.occasions-1) # Annual number of newly marked
# Determine annual survival probabilities
mean.phi <- 0.65
var.phi <- 1 # Temporal variance of survival
p <- rep(0.4, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked), byrow = TRUE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```
# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

In the BUGS model description, we only alter parts in the “Priors and constraints” sections; no change is required in the likelihood part. In particular, we have to implement the random-effects formulation (formula above). The prior choices for  $\mu$  and for  $\sigma^2$  need some thought. Because  $\mu$  is the mean survival on the logit scale, a noninformative prior on the logit scale would be a normal distribution with a wide variance. Yet, this prior will not be noninformative on the probability scale. In the code below, we provide two options: first, a normal distribution with a wide variance for  $\mu$ , and second, a uniform distribution for  $\text{logit}^{-1}(\mu)$ , which is noninformative on the probability scale but informative on the logit scale. A prior is also needed for  $\sigma^2$ . Following Gelman (2006), we use a uniform distribution for the standard deviation because this induces little information. We will fit the same model under which we generated the data, that is, model  $\phi_{\underline{t}}, p.$ , where by the underlined index for time, we denote random time effects.

```
# Specify model in BUGS language
sink("cjs-temp-raneff.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
    for (t in f[i] : (n.occasions-1)) {
        logit(phi[i,t]) <- mu + epsilon[t]
        p[i,t] <- mean.p
    } #t
} #i
for (t in 1: (n.occasions-1)) {
    epsilon[t] ~ dnorm(0, tau)
}

#mu ~ dnorm(0, 0.001)                      # Prior for logit of mean survival
#mean.phi <- 1 / (1+exp(-mu))              # Logit transformation
mean.phi ~ dunif(0, 1)                      # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi))         # Logit transformation
sigma ~ dunif(0, 10)                         # Prior for standard deviation
tau <- pow(sigma, -2)                        # Temporal variance
sigma2 <- pow(sigma, 2)                      # Prior for mean recapture
mean.p ~ dunif(0, 1)                         # Prior for mean recapture

# Likelihood
for (i in 1:nind) {
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1) : n.occasions) {
```

```

# State process
z[i,t] ~ dbern(mu1[i,t])
mu1[i,t] <- phi[i,t-1] * z[i,t-1]
# Observation process
y[i,t] ~ dbern(mu2[i,t])
mu2[i,t] <- p[i,t-1] * z[i,t]
} #t
} #i
}, fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
z = known.state.cjs(CH))

# Initial values
inits <- function() {list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
sigma = runif(1, 0, 10), mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p", "sigma2")

# MCMC settings
ni <- 10000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 17 min)
cjs.ran <- bugs(bugs.data, inits, parameters, "cjs-temp-raneff.bug",
n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
bugs.directory = bugs.dir, working.directory = getwd())

```

The chains evolve slowly and convergence is not achieved swiftly. This can be improved if a smaller range is chosen for the prior for the standard deviation of the temporal variance. Here, we used a uniform prior in the interval between 0 and 10, thus the variance could take values between 0 and 100. Had we chosen a higher upper bound, the estimate would probably not change (recall we simulated the data with a variance of 1), but the chains would converge even more slowly. Computation for this model is more efficient for the multinomial formulation of the model (see Section 7.10).

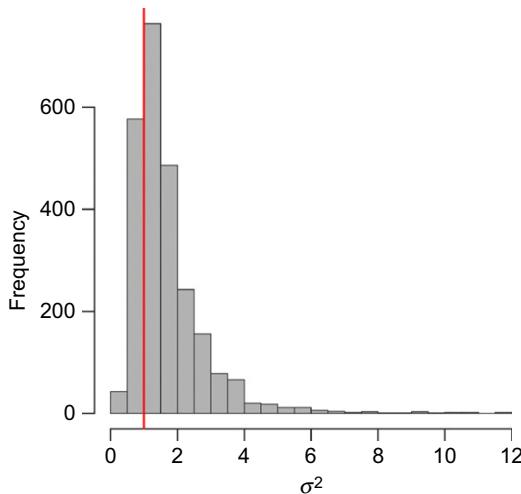
```

# Summarize posteriors
print(cjs.ran, digits = 3)

      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat   n.eff
mean.phi 0.634 0.073 0.488 0.590 0.634 0.678 0.787 1.017    120
mean.p   0.394 0.024 0.350 0.378 0.394 0.411 0.441 1.001   2000
sigma2   1.700 1.152 0.548 1.006 1.402 2.005 4.687 1.006    440

# Produce histogram
hist(cjs.ran$sims.list$sigma2, col = "gray", nclass = 35, las = 1,
xlab = expression(sigma^2), main = "")
abline(v = var.phi, col = "red", lwd = 2)

```



**FIGURE 7.3** Posterior distribution of the temporal variance in apparent survival (red: value used for data generation).

The histogram of the posterior samples of the temporal variance for one simulated data set is shown in Fig. 7.3. The point estimate may seem biased; however, recall that this is just a single simulation, and we would need many simulations to check for any bias (see exercise 4 in Section 7.13).

### 7.4.3 Temporal Covariates

Often we are not only interested in getting a point estimate of survival or of its temporal variability, but also in identifying factors affecting survival. One way to do this is to see whether the observed temporal pattern in survival matches the temporal variation of an environmental factor (e.g., winter severity). From a nonzero correlation we would then infer an effect of that factor on survival. However, regardless of how the model is specified, such evidence is of correlative nature, thus causation cannot be inferred. A properly designed experiment is needed to infer causation, which is not easy in population studies (but see Schwarz, 2002).

Traditionally, so-called ultrastructural modeling has been used to model survival as a function of a covariate ( $x$ ) (Lebreton et al., 1992; Link, 1999):

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_t.$$

This model assumes that the entire temporal variability of survival could be explained by the covariate  $x$ ; it is analogous to a linear regression model without residuals. This seems quite unrealistic, but in earlier times, this was the only way that the relationship between survival and a covariate could be modeled. A more realistic approach is to assume that only part of the temporal variability of survival is explained by the covariate, another part being unexplained random variation. Thus, we specify this model:

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_t + \varepsilon_t$$

$$\varepsilon_t \sim \text{Normal}(0, \sigma^2).$$

The residual variance ( $\sigma^2$ ) is the unexplained temporal variance. This allows us to estimate the amount of the total temporal variance which is explained by covariate  $x$ . We need to fit a model without the covariate to get an estimate of the total temporal variance ( $\sigma_{\text{total}}^2$ ). The proportion of the variance explained by covariate  $x$  is then  $(\sigma_{\text{total}}^2 - \sigma^2)/\sigma_{\text{total}}^2$  (Grosbois et al., 2008).

To illustrate the model with the little owl example, we assume a mean survival of 0.65 and a negative effect of winter severity with logistic-linear slope of  $-0.3$ . The winter severity index is standardized (mean = 0, variance = 1) and the residual temporal variance not explained by winter severity has variance of 0.2.

```
# Define parameter values
n.occasions <- 20 # Number of capture occasions
marked <- rep(15, n.occasions-1) # Annual number of newly marked
# individuals
mean.phi <- 0.65
p <- rep(0.4, n.occasions-1)
beta <- -0.3 # Slope of survival-winter
# relationship
r.var <- 0.2 # Residual temporal variance

# Draw annual survival probabilities
winter <- rnorm(n.occasions-1, 0, 1^0.5)
logit.phi <- qlogis(mean.phi) + beta*winter + rnorm(n.occasions-1, 0,
r.var^0.5)
phi <- plogis(logit.phi)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked),
byrow = TRUE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```

# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Specify model in BUGS language
sink("cjs-cov-ranef.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)){
    logit(phi[i,t]) <- mu + beta*x[t] + epsilon[t]
    p[i,t] <- mean.p
    } #t
  } #i
for (t in 1:(n.occasions-1)){
  epsilon[t] ~ dnorm(0, tau)
  phi.est[t] <- 1 / (1+exp(-mu-beta*x[t]-epsilon[t])) # Yearly
               survival
}
mu ~ dnorm(0, 0.001)                                # Prior for logit of mean survival
mean.phi <- 1 / (1+exp(-mu))                         # Logit transformation
beta ~ dnorm(0, 0.001)I(-10, 10)                      # Prior for slope parameter
sigma ~ dunif(0, 10)                                    # Prior on standard deviation
tau <- pow(sigma, -2)                                 # Residual temporal variance
sigma2 <- pow(sigma, 2)                               # Prior for mean recapture
mean.p ~ dunif(0, 1)                                    # Prior for mean recapture

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions){
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
    } #t
  } #i
}

",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
z = known.state.cjs(CH), x = winter)

```

```

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mu = rnorm(1), sigma =
  runif(1, 0, 5), beta = runif(1, -5, 5), mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "mean.p", "phi.est", "sigma2", "beta")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 12 min)
cjs.cov <- bugs(bugs.data, inits, parameters, "cjs-cov-raneff.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

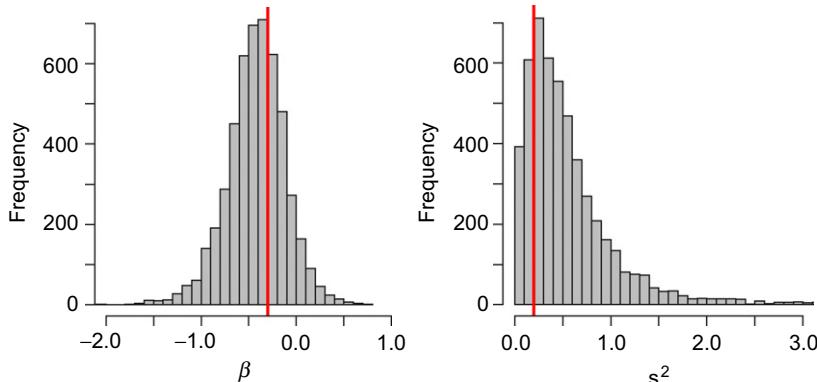
In general, models with random effects are more difficult to fit, and we need to run long Markov chains to achieve satisfactory convergence for all parameters. The posterior distributions of the slope and the residual environmental variation (temporal variation not explained by the covariate) are shown in Fig. 7.4.

```

# Summarize posteriors
print(cjs.cov, digits = 3)

      mean     sd   2.5%    25%    50%    75% 97.5%   Rhat n.eff
mean.phi    0.707  0.050  0.611  0.676  0.705  0.736  0.805 1.012 1400
mean.p      0.403  0.032  0.343  0.381  0.403  0.424  0.467 1.002 1800
phi.est[1]   0.686  0.121  0.423  0.610  0.696  0.769  0.902 1.003 1200
[ ... ]

```



**FIGURE 7.4** Posterior distributions of the covariate effect (slope parameter  $\beta$ ) and of environmental variability (the residual temporal variance  $\sigma^2$ ). Red lines indicate the values used for simulating the data.

```

phi.est[19]  0.681  0.120  0.427  0.603  0.690  0.764  0.902 1.003   950
sigma2       0.566  0.541  0.047  0.234  0.426  0.714  2.012 1.008   390
beta        -0.422  0.308 -1.080 -0.600 -0.410 -0.226  0.160 1.006  1400

# Produce graph
par(mfrow = c(1, 2), las = 1)
hist(cjs.cov$sims.list$beta, nclass = 25, col = "gray", main = "",
     xlab = expression(beta), ylab = "Frequency")
abline(v = -0.3, col = "red", lwd = 2)
hist(cjs.cov$sims.list$sigma2, nclass = 50, col = "gray", main = "",
     xlab = expression(sigma^2), ylab = "Frequency", xlim=c(0, 3))
abline(v = 0.2, col = "red", lwd = 2)

```

## 7.5 MODELS WITH INDIVIDUAL VARIATION

So far we have modeled temporal effects, assuming identical survival and recapture for all individuals. Now, we relax this assumption and model individual heterogeneity. Thus, we model effects along the row axis of the capture-history matrix. As with models for time effects, we can model individual effects in different ways—individual effects can be categorical or continuous, fixed or random, or latent or explained by measured covariates (compare with model M<sub>h</sub> in Section 6.2). Moreover, individual effects can be constant over time, or they may change over time. Here, we only consider the simpler case, where they are constant over time.

### 7.5.1 Fixed Group Effects

We may specify fixed effects when we are interested in the estimates of particular groups (e.g., sex) and if the number of groups is low, as it is difficult to estimate the between-group variance with a small number of groups. We define  $g_i$  as a categorical variable with G levels (i.e., number of groups) indicating the group membership. The model for survival with a fixed group effect is

$$\phi_{i,t} = \beta_{g(i)},$$

where index  $g(i)$  denotes the group  $g$  to which individual  $i$  belongs and  $\beta_g$  ( $g = 1 \dots G$ ) are the estimated fixed group effects. Because there are no other effects in the model, we can model directly on the probability scale, and the prior distribution ensures that the parameter estimates are between 0 and 1.

We illustrate the model to estimate sex-specific survival and recapture probabilities with simulated data of little owls. We first simulate two separate capture–recapture data sets; one for males and another for females.

Then we create the grouping variable  $g$  (named “group”) and merge the two capture–recapture data sets.

```
# Define parameter values
n.occasions <- 12 # Number of capture occasions
marked <- rep(30, n.occasions-1) # Annual number of newly marked
# individuals
phi.f <- rep(0.65, n.occasions-1) # Survival of females
p.f <- rep(0.6, n.occasions-1) # Recapture prob. of females
phi.m <- rep(0.8, n.occasions-1) # Survival of males
p.m <- rep(0.3, n.occasions-1) # Recapture prob. of males

# Define matrices with survival and recapture probabilities
PHI.F <- matrix(phi.f, ncol = n.occasions-1, nrow = sum(marked))
P.F <- matrix(p.f, ncol = n.occasions-1, nrow = sum(marked))
PHI.M <- matrix(phi.m, ncol = n.occasions-1, nrow = sum(marked))
P.M <- matrix(p.m, ncol = n.occasions-1, nrow = sum(marked))

# Simulate capture-histories
CH.F <- simul.cjs(PHI.F, P.F, marked)
CH.M <- simul.cjs(PHI.M, P.M, marked)

# Merge capture-histories by row
CH <- rbind(CH.F, CH.M)

# Create group variable
group <- c(rep(1, dim(CH.F)[1]), rep(2, dim(CH.M)[1]))

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

Finally, we write the model in BUGS language and fit it to the data.

```
# Specify model in BUGS language
sink("cjs-group.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
    for (t in f[i]:(n.occasions-1)) {
        phi[i,t] <- phi.g[group[i]]
        p[i,t] <- p.g[group[i]]
    } #
} #
for (u in 1:g) {
    phi.g[u] ~ dunif(0, 1) # Priors for group-specific
                           # survival
    p.g[u] ~ dunif(0, 1) # Priors for group-specific
                           # recapture
}

# Likelihood
for (i in 1:nind) {
```

```

# Define latent state at first capture
z[i,f[i]] <- 1
for (t in (f[i]+1):n.occasions) {
  # State process
  z[i,t] ~ dbern(mul[i,t])
  mul[i,t] <- phi[i,t-1] * z[i,t-1]
  # Observation process
  y[i,t] ~ dbern(mu2[i,t])
  mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
},
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions = dim(CH)[2],
z = known.state.cjs(CH), g = length(unique(group)), group = group)

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), phi.g = runif(length(unique(group)), 0, 1),
p.g = runif(length(unique(group)), 0, 1))}

# Parameters monitored
parameters <- c("phi.g", "p.g")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 2 min)
cjs.group <- bugs(bugs.data, inits, parameters, "cjs-group.bug",
n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
bugs.directory = bugs.dir, working.directory = getwd())

```

The parameter estimates are close to the values used to generate the data.

```

# Summarize posteriors
print(cjs.group, digits = 3)

      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
phi.g[1] 0.656  0.020  0.617  0.642  0.656  0.669  0.694 1.001  3000
phi.g[2] 0.796  0.018  0.760  0.784  0.796  0.808  0.831 1.002  1900
p.g[1]   0.599  0.029  0.541  0.578  0.599  0.619  0.654 1.002  1700
p.g[2]   0.325  0.021  0.285  0.311  0.324  0.339  0.368 1.002  1900

```

## 7.5.2 Random Group Effects

We may specify random group effects when we are interested in an overall mean and the variability between groups. A typical example of random group effects is provided by local populations, where we are

interested in estimating spatial variation of survival (Grosbois et al., 2009). Survival is then modeled as

$$\text{logit}(\phi_{i,t}) = \beta_{g(i)} \\ \beta_g \sim \text{Normal}(\bar{\beta}, \sigma^2),$$

where  $\sigma^2$  is the variance of logit survival between groups,  $\beta_g$  are the random group effects, and  $\bar{\beta}$  is the overall mean. Note that we now use the logit link function to ensure that the realized group-specific survival probabilities ( $\text{logit}^{-1}(\beta_g)$ ) are bound in the interval [0, 1].

Because most BUGS code is identical to that in Section 7.5.1, we just show the part which needs modification:

```
# Priors and constraints
for (i in 1:nind) {
    for (t in f[i]:(n.occasions-1)) {
        logit(phi[i,t]) <- beta[group[i]]
        p[i,t] <- mean.p
    } #t
} #i
for (u in 1:g) {
    beta[u] ~ dnorm(mean.beta, tau)
    phi.g[u] <- 1 / (1+exp(-beta[u])) # Back-transformed
                                         group-specific survival
}
mean.beta ~ dnorm(0, 0.001)           # Prior for logit of mean survival
mean.phi <- 1 / (1+exp(-mean.beta)) # Back-transformed mean survival
sigma ~ dunif(0, 10)                 # Prior for sd of logit of survival
                                         variability
tau <- pow(sigma, -2)
mean.p ~ dunif(0, 1)                  # Prior for mean recapture
```

### 7.5.3 Individual Random Effects

As an extreme case of a random group effect, we could also consider each individual as belonging to its own group. This model would not be identifiable when groups are treated as fixed affects, but it is when we treat groups as random effects. Conceptually, we imagine that there is an average survival, around which there is individual-specific noise. To specify individual random effects, we write

$$\text{logit}(\phi_{i,t}) = \mu + \varepsilon_i \\ \varepsilon_i \sim \text{Normal}(0, \sigma^2),$$

where  $\sigma^2$  is the variance of logit survival among individuals, and  $\mu$  is the overall mean logit survival. The interest of an analysis with individual random effects may be in estimating the mean, the variance, or even the realized survival “residuals” of each individual (sometimes called

"frailty"; Cam et al., 2002). Such a model also provides the base for modeling survival as a function of an individual covariate  $x_i$  (e.g., size of an individual). This model can be written as

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_i + \varepsilon_i \\ \varepsilon_i \sim \text{Normal}(0, \sigma^2),$$

where  $\beta$  is the slope of covariate  $x$  on logit survival.

Models with random individual variation in survival are particularly important for the study of senescence (Cam et al., 2002). If individual variation is not included, senescence could easily be overlooked because a decline with age may be offset by increasing proportions of high-quality individuals in the population (Service, 2000; van de Pol and Verhulst, 2006). Sometimes, such a model may also be adopted for recapture probabilities because they are likely to differ among individuals in a similar manner.

Capture–recapture data are often subject to overdispersion, which may be due to a lack of independence among individuals (Lebreton et al., 1992). Overdispersion can be detected with a goodness-of-fit test (Lebreton et al., 1992; Choquet et al., 2001). If overdispersion is not corrected for, parameter estimators tend to be unbiased, but their variances (e.g., standard errors) will be too small (Anderson et al., 1994). The frequentist solution is to calculate a variance inflation factor from the goodness-of-fit test that is called c-hat in the capture–recapture literature, and to compute the true variance of the estimates as the product of the apparent variance and c-hat. An analogous Bayesian solution is to use a model with individual random effects (see also chapter 14 in Kéry 2010 and Section 4.2). The advantage of the Bayesian solution is the flexibility in specifying lack of independence in either survival only, recapture only, or in both parameters.

We now simulate little owl data and analyze them. We assume mean survival of 0.65 and individual variability with a variance of 0.5.

```
# Define parameter values
n.occasions <- 20 # Number of capture occasions
marked <- rep(30, n.occasions-1) # Annual number of newly marked
                                individuals
mean.phi <- 0.65
p <- rep(0.4, n.occasions-1)
v.ind <- 0.5

# Draw annual survival probabilities
logit.phi <- rnorm(sum(marked), qlogis(mean.phi), v.ind^0.5)
phi <- plogis(logit.phi)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked),
               byrow = FALSE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked))
```

```

# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Specify model in BUGS language
sink("cjs-ind-ranef.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        logit(phi[i,t]) <- mu + epsilon[i]
        p[i,t] <- mean.p
    } #t
} #i
for (i in 1:nind){
    epsilon[i] ~ dnorm(0, tau)
}
mean.phi ~ dunif(0, 1)           # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi)) # Logit transformation
sigma ~ dunif(0, 5)              # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)
mean.p ~ dunif(0, 1)             # Prior for mean recapture

# Likelihood
for (i in 1:nind){
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions){
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
        mu2[i,t] <- p[i,t-1] * z[i,t]
    } #t
} #i
}

",fill = TRUE)
sink()

# Bundle data
bugs.data<- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
z = known.state.cjs(CH))

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0, 1),
mean.p = runif(1, 0, 1), sigma = runif(1, 0, 2))}
```

```

# Parameters monitored
parameters <- c("mean.phi", "mean.p", "sigma2")

# MCMC settings
ni <- 50000
nt <- 6
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT 73 min)
cjs.ind <- bugs(bugs.data, inits, parameters, "cjs-ind-raneff.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

We need relatively long runs to reach satisfactory convergence. We note that the random-effects distribution could also be truncated like  $\epsilon_{i,j} \sim \text{dnorm}(0, \tau^2) I(-15, 15)$  to improve mixing of the chains (see Appendix 1, tipp 16). The posterior distributions of the two parameters, mean survival and variability among individuals, show good agreement with the simulated parameters (Fig. 7.5).

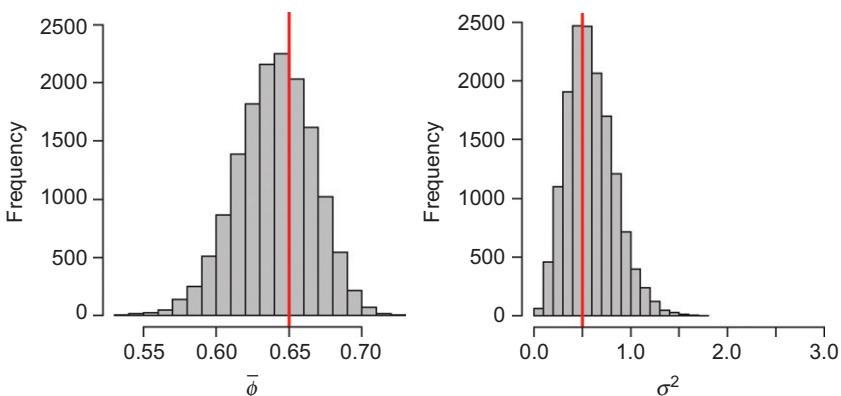
```

# Summarize posteriors
print(cjs.ind, digits = 3)

      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
mean.phi 0.640 0.026 0.587 0.623 0.641 0.658 0.688 1.001 15000
mean.p   0.410 0.021 0.368 0.396 0.410 0.424 0.451 1.001 13000
sigma2   0.586 0.244 0.176 0.410 0.560 0.739 1.132 1.012 1800

# Produce graph
par(mfrow = c(1, 2), las = 1)
hist(cjs.ind$sims.list$mean.phi, nclass = 25, col = "gray", main = "",
  xlab = expression(bar(phi)), ylab = "Frequency")
abline(v = mean.phi, col = "red", lwd = 2)

```



**FIGURE 7.5** Posterior distributions of mean survival and of the individual variance in survival. Red lines indicate the values used for data simulation.

```
hist(cjs.ind$sims.list$sigma2, nclass = 15, col = "gray", main = "",
     xlab = expression(sigma^2), ylab = "Frequency", xlim = c(0, 3))
abline(v = v.ind, col = "red", lwd = 2)
```

If we wanted to estimate survival as a function of an individual covariate  $x$ , then we just have to adapt a small part in the code:

```
# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)){
    logit(phi[i,t]) <- mu + beta*x[i] + epsilon[i]
    p[i,t] <- mean.p
  } #t
} #i
for (i in 1:nind) {
  epsilon[i] ~ dnorm(0, tau)
}
mean.phi ~ dunif(0, 1)           # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi)) # Logit transformation
beta ~ dnorm(0, 0.001)           # Prior for covariate slope
sigma ~ dunif(0, 5)              # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)
mean.p ~ dunif(0, 1)             # Prior for mean recapture
```

Of course, we also have to give initial values for the new stochastic node beta, to include the covariate  $x$  in bugs.data, and to monitor beta.

Individual covariates may also change over time, such as, for example, body mass. The difficulty is that the covariate is unknown at occasions when the individual was not captured. Estimating the effects of individual time-varying covariates on survival is a challenge and different approaches have been proposed (Bonner and Schwarz, 2006; Catchpole et al., 2008; King et al., 2010).

## 7.6 MODELS WITH TIME AND GROUP EFFECTS

### 7.6.1 Fixed Group and Time Effects

Clearly we can combine the two concepts introduced in Sections 7.4 and 7.5 and model structure both along the time and along the individual axis of the capture-history matrix. The changes needed in the model code are merely an explicit GLM formulation of effects. This offers great flexibility as we can consider interacting or additive time and group effects, and we can treat either or both as random. The different combinations are straightforward and easy to implement, so we now focus in detail on one particular model that is often used, an additive model with fixed time and group effects.

Consider two groups of individuals (e.g., males and females) whose survival varies in parallel over time. Denoting sex by  $g$  (for group) and time by  $t$ , we can call this model  $\{\phi_{g+tr}, p_g\}$ . Using the GLM formulation, we specify the survival model as

$$\text{logit}(\phi_{i,t}) = \beta_{g(i)} + \gamma_t,$$

where  $\beta_g$  is the effect of the sex  $g$  of individual  $i$  and  $\gamma_t$  are the fixed time effects. Written in this way, the model is overparameterized. We must either specify the  $\beta_g$  as the survival probabilities of the first year, and thus set  $\gamma_1 = 0$ , or we specify that  $\gamma_t$  are the survival probabilities of the first group and set  $\beta_1 = 0$ . Consequently,  $\beta_2$  is then the difference in survival between the first and the second group. Such constraints must be specified in the BUGS model code, and are usually called corner constraints (Ntzoufras, 2009; Kéry, 2010).

For the simulation example, we assume constant recapture probabilities that are higher for females than for males. We simulate two capture-history data sets, one for males and one for females, merge them, create a group variable, and finally fit the model.

```
# Define parameter values
n.occasions <- 12 # Number of capture occasions
marked <- rep(50, n.occasions-1) # Annual number of newly marked
# individuals
phi.f <- c(0.6, 0.5, 0.55, 0.6, 0.5, 0.4, 0.6, 0.5, 0.55, 0.6, 0.7)
p.f <- rep(0.6, n.occasions-1)
diff <- 0.5 # Difference between male and female survival on logit
# scale
phi.m <- plogis(qlogis(phi.f) + diff)
p.m <- rep(0.3, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI.F <- matrix(rep(phi.f, sum(marked)), ncol = n.occasions-1,
  nrow = sum(marked), byrow = TRUE)
P.F <- matrix(rep(p.f, sum(marked)), ncol = n.occasions-1,
  nrow = sum(marked), byrow = TRUE)
PHI.M <- matrix(rep(phi.m, sum(marked)), ncol = n.occasions-1,
  nrow = sum(marked), byrow = TRUE)
P.M <- matrix(rep(p.m, sum(marked)), ncol = n.occasions-1,
  nrow = sum(marked), byrow = TRUE)

# Simulate capture-histories
CH.F <- simul.cjs(PHI.F, P.F, marked)
CH.M <- simul.cjs(PHI.M, P.M, marked)

# Merge capture-histories
CH <- rbind(CH.F, CH.M)

# Create group variable
group <- c(rep(1, dim(CH.F)[1]), rep(2, dim(CH.M)[1]))
```

```
# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```

The next piece of code writes the model in BUGS language, and the remaining R code fits the model:

```
# Specify model in BUGS language
sink("cjs-add.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind){
  for (t in f[i]:(n.occasions-1)){
    logit(phi[i,t]) <- beta[group[i]] + gamma[t]
    p[i,t] <- p.g[group[i]]
  } #t
} #i
# for survival parameters
for (t in 1:(n.occasions-1)){
  gamma[t] ~ dnorm(0, 0.01)I(-10, 10)          # Priors for time
                                                    effects
  phi.g1[t] <- 1 / (1+exp(-gamma[t]))           # Back-transformed
                                                    survival of males
  phi.g2[t] <- 1 / (1+exp(-gamma[t]-beta[2])) # Back-transformed
                                                    survival of females
}
beta[1] <- 0                                     # Corner constraint
beta[2] ~ dnorm(0, 0.01)I(-10, 10)             # Prior for difference in male
                                                    and female survival
# for recapture parameters
for (u in 1:g){
  p.g[u] ~ dunif(0, 1)                          # Priors for group-spec.
                                                    recapture
}

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions){
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
}
```

```

  ",fill = TRUE)
sink()

# Bundle data
bugs.data<- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
  z = known.state.cjs(CH), g = length(unique(group)), group = group)

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), gamma =
  rnorm(n.occasions-1), beta = c(NA, rnorm(1)), p.g = runif(length
  (unique(group)), 0, 1))}

# Parameters monitored
parameters <- c("phi.g1", "phi.g2", "p.g", "beta")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

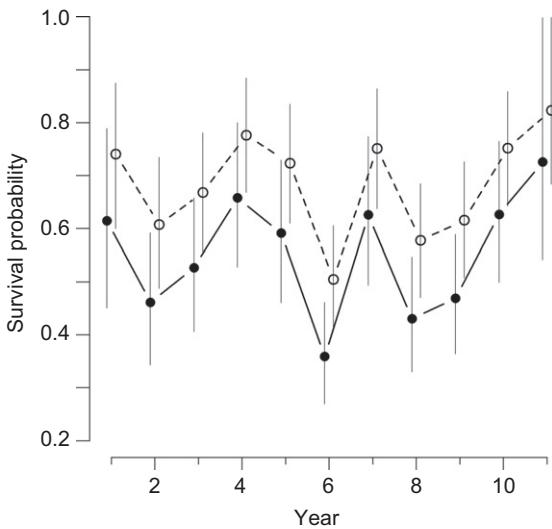
# Call WinBUGS from R (BRT 7 min)
cjs.add <- bugs(bugs.data, inits, parameters, "cjs-add.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

# Summarize posteriors
print(cjs.add, digits = 3)

      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
phi.g1[1] 0.614 0.088 0.451 0.554 0.611 0.672 0.789 1.002 2800
phi.g1[2] 0.461 0.065 0.343 0.416 0.459 0.504 0.592 1.001 2800
[...]
phi.g2[10] 0.752 0.055 0.642 0.716 0.753 0.790 0.859 1.010 260
phi.g2[11] 0.823 0.079 0.683 0.770 0.818 0.868 0.999 1.030 90
p.g[1]     0.567 0.034 0.499 0.545 0.567 0.590 0.633 1.006 350
p.g[2]     0.318 0.022 0.277 0.302 0.317 0.333 0.361 1.005 450
beta[2]    0.603 0.127 0.360 0.515 0.605 0.687 0.848 1.008 300

# Figure of male and female survival
lower.f <- upper.f <- lower.m <- upper.m <- numeric()
for (t in 1:(n.occasions-1)){
  lower.f[t] <- quantile(cjs.add$sims.list$phi.g1[,t], 0.025)
  upper.f[t] <- quantile(cjs.add$sims.list$phi.g1[,t], 0.975)
  lower.m[t] <- quantile(cjs.add$sims.list$phi.g2[,t], 0.025)
  upper.m[t] <- quantile(cjs.add$sims.list$phi.g2[,t], 0.975)
}
plot(x=(1:(n.occasions-1))-0.1, y = cjs.add$mean$phi.g1, type = "b",
  pch = 16, ylim = c(0.2, 1), ylab = "Survival probability",
  xlab = "Year", bty = "n", cex = 1.5, axes = FALSE)
axis(1, at = 1:11, labels = rep(NA,11), tcl = -0.25)
axis(1, at = seq(2,10,2), labels = c("2","4","6","8","10"))
axis(2, at = seq(0.2, 1, 0.1), labels = c("0.2", NA, "0.4", NA, "0.6", NA,
  "0.8", NA, "1.0"), las = 1)
segments((1:(n.occasions-1))-0.1, lower.f, (1:(n.occasions-1))-0.1,
  upper.f)

```



**FIGURE 7.6** Posterior means (with 95% CRIs) of male (open circles) and female survival (closed symbols) under the additive model.

```
points(x = (1:(n.occasions-1))+0.1, y = cjs.add$mean$phi.g2,
       type = "b", pch = 1, lty = 2, cex = 1.5)
segments((1:(n.occasions-1))+0.1, lower.m, (1:(n.occasions-1))+0.1,
         upper.m)
```

The posterior means of male and female survival estimated under the additive model are shown in Fig. 7.6. Survival of the two sexes varies in parallel over time, but on the logit scale. Hence, on the probability scale the two curves are not parallel—as the difference becomes smaller the closer the estimates are to 1 or 0.

To fit a model with an interaction between sex and time (i.e., survival of each sex varies independently from each other over time), we would change the “Priors and constraints” part of the model as follows:

```
# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)) {
    phi[i,t] <- eta.phi[group[i],t]
    p[i,t] <- p.g[group[i]]
  } #t
} #i
# for survival parameters
for (u in 1:g) {
  for (t in 1:(n.occasions-1)) {
    eta.phi[u,t] ~ dunif(0, 1) # Prior for time and group-spec.
                                survival
```

```

        } #t
    } #g
# for recapture parameters
for (u in 1:g){
    p.g[u] ~ dunif(0, 1)                      # Priors for group-spec. recapture
}

```

## 7.6.2 Fixed Group and Random Time Effects

We may combine fixed group and random time effects to estimate temporal variability of survival (or recapture) in each group separately. As for the interacting model before, such a model would assume that the temporal variability of each group is independent of that in the other group(s).

$$\text{logit}(\phi_{i,t}) = \mu_{g(i)} + \varepsilon_{g(i),t}$$

$$\varepsilon_{g,t} \sim \text{Normal}(0, \sigma_g^2),$$

where  $\mu_g$  are the group-specific means and  $\sigma_g^2$  the group-specific temporal variances. The model code again only needs changes to the "Priors and constraints" part and looks like:

```

# Priors and constraints
for (i in 1:nind){
    for (t in f[i]:(n.occasions-1)){
        logit(phi[i,t]) <- eta.phi[group[i],t]
        p[i,t] <- p.g[group[i]]
        } #t
    } #i
# for survival parameters
for (u in 1:g){
    for (t in 1:(n.occasions-1)){
        eta.phi[u,t] <- mu.phi[u] + epsilon[u,t]
        epsilon[u,t] ~ dnorm(0, tau[u])
        } #t
    mean.phi[u] ~ dunif(0, 1)                  # Priors on mean group-spec.
                                                    survival
    mu.phi[u] <- log(mean.phi[u] / (1-mean.phi[u]))
    sigma[u] ~ dunif(0, 10)                    # Priors for group-spec. sd
    tau[u] <- pow(sigma[u], -2)
    sigma2[u] <- pow(sigma[u], 2)
    } #g
# for recapture parameters
for (u in 1:g){
    p.g[u] ~ dunif(0,1)                      # Priors for group-spec.
                                                    recapture
}

```

An alternative way to write the same model is to treat the residuals as a realization from a multivariate normal distribution

$$\varepsilon_{g,t} \sim \text{MVN}(0, \Sigma_{g,t}),$$

where  $\Sigma_{g,t}$  is the variance–covariance matrix that describes the temporal variance of and the temporal covariance among groups. As we assume independence among groups, the covariance between groups is zero, and the matrix for two groups is as follows:

$$\Sigma_{g,t} = \begin{pmatrix} \sigma_{g1}^2 & 0 \\ 0 & \sigma_{g2}^2 \end{pmatrix}.$$

Temporal variability in survival is usually induced by environmental factors (e.g., weather, food supply). As such, we do not expect survival of groups of individuals from the same population (e.g., sexes or age classes) to vary independently over time. Therefore, we may want to fit a sort of additive model, but where the temporal variance is treated as random. This can be done by considering a correlation of the temporal variability of each group, that is treating two sets of random effects as correlated (Link and Barker, 2005). The advantage of such a model is that (1) the temporal correlation is interpretable as a biological parameter (the extent to which survival varies in common among groups) and (2) the estimates of temporal variability become more precise because information is shared among groups. The temporal correlation of parameters also needs to be included in stochastic population models. Generally, the population growth rate becomes smaller with increasing positive correlation between survival parameters (Caswell, 2001).

With two groups, this model is written as follows:

$$\begin{aligned} \text{logit}(\phi_{i,t}) &= \mu_{g(i)} + \varepsilon_{g(i),t} \\ \varepsilon_{g,t} &\sim \text{MVN}(0, \Sigma_{g,t}) \\ \Sigma_{g,t} &= \begin{pmatrix} \sigma_{g1}^2 & \rho\sigma_{g1}\sigma_{g2} \\ \rho\sigma_{g1}\sigma_{g2} & \sigma_{g2}^2 \end{pmatrix}, \end{aligned}$$

where  $\rho$  is the temporal correlation coefficient between the two groups. Note that the correlation coefficient between two variables  $g_1$  and  $g_2$  is

$$\rho = \frac{\text{cov}(g1, g2)}{\sqrt{\sigma_{g1}^2 \sigma_{g2}^2}}, \text{ and thus } \Sigma_{g,t} \text{ could also be written as}$$

$$\Sigma_{g,t} = \begin{pmatrix} \sigma_{g1}^2 & \text{cov}(g1, g2) \\ \text{cov}(g1, g2) & \sigma_{g2}^2 \end{pmatrix}.$$

Estimating correlation coefficients (or covariances) is challenging, in particular, if there are more than two parameters. This is because several conditions must be met. For example, all correlations must be in the interval  $-1$  and  $1$ , and they are jointly constrained in a complicated way. A standard choice for the prior of the elements of matrix  $\Sigma$  is the inverse

Wishart distribution, which ensures that the estimated parameters have the desired properties.

The inverse Wishart distribution ( $IW(R, df)$ ) has two parameters: the scale matrix ( $R$ ), with dimension  $K \times K$  for  $K$  modeled parameters, and the degrees of freedom ( $df$ ). Depending on the choice of these parameters, we incorporate into the analysis prior information about the correlation coefficients or about the variances (Link and Barker, 2005; Gelman and Hill, 2007). For a uniform prior on the correlation coefficients, we must fix  $df = K + 1$ . The values of the scale matrix  $R$  have an effect on the priors for the variances; large values of  $R$  set the prior means of the variances to large values. Because the specification of the priors for matrix  $\Sigma$  is difficult, we recommend conducting sensitivity analyses. The BUGS code to fit this model is as follows:

```
# Specify model in BUGS language
sink("cjs-temp-corr.bug")
cat(
model {

# Priors and constraints
for (i in 1:nind) {
    for (t in f[i]:(n.occasions-1)) {
        logit(phi[i,t]) <- eta.phi[t,group[i]]
        p[i,t] <- p.g[group[i]]
    } #
} #i
# for survival parameters
for (t in 1:(n.occasions-1)) {
    eta.phi[t,1:g] ~ dmmnorm(mu.phi[], Omega[,])
} #
for (u in 1:g) {
    mean.phi[u] ~ dunif(0, 1)      # Priors on mean group-spec. survival
    mu.phi[u] <- log(mean.phi[u] / (1-mean.phi[u]))
} #
Omega[1:g, 1:g] ~ dwish(R[,], df) # Priors for variance-covariance
                                    matrix
Sigma[1:g, 1:g] <- inverse(Omega[,])

# for recapture parameters
for (u in 1:g) {
    p.g[u] ~ dunif(0, 1)          # Priors for group-spec. recapture
}

# Likelihood
for (i in 1:nind) {
    # Define latent state at first capture
    z[i,f[i]] <- 1
    for (t in (f[i]+1):n.occasions) {
        # State process
        z[i,t] ~ dbern(mu1[i,t])
        mu1[i,t] <- phi[i,t-1] * z[i,t-1]
        # Observation process
        y[i,t] ~ dbern(mu2[i,t])
    }
}
```

```

    mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
}
",fill = TRUE)
sink()

```

The parameters of the inverse Wishart distribution ( $R, df$ ) are provided as data. Here, we choose parameters that result in an uninformative prior for the correlation.

```

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
  z = known.state.cjs(CH), g = length(unique(group)), group = group,
  R = matrix(c(5, 0, 0, 1), ncol = 2), df = 3)

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), p.g = runif(length
  (unique(group)), 0, 1), Omega = matrix(c(1, 0, 0, 1), ncol = 2))}

# Parameters monitored
parameters <- c("eta.phi", "p.g", "Sigma", "mean.phi")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 5 min)
cjs.corr <- bugs(bugs.data, inits, parameters, "cjs-temp-corr.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

# Summarize posteriors
print(cjs.corr, digits = 3)

  mean      sd   2.5%    25%    50%    75% 97.5%   Rhat n.eff
eta.phi[1,1] 0.457 0.391 -0.257  0.190  0.434  0.688 1.304 1.003     870
eta.phi[1,2] 0.794 0.384  0.103  0.537  0.770  1.020 1.605 1.001    3000
[ ... ]
eta.phi[11,1] 0.945 0.445  0.219  0.647  0.892  1.194 1.995 1.010     420
eta.phi[11,2] 0.800 0.363  0.165  0.554  0.762  1.031 1.546 1.002    1100
p.g[1]        0.572 0.032  0.511  0.550  0.572  0.594 0.636 1.002    1600
p.g[2]        0.327 0.023  0.283  0.311  0.327  0.343 0.375 1.003     970
Sigma[1,1]    0.790 0.391  0.323  0.523  0.704  0.957 1.793 1.001    3000
Sigma[1,2]    0.073 0.156 -0.197 -0.014  0.057  0.146 0.440 1.003    3000
Sigma[2,1]    0.073 0.156 -0.197 -0.014  0.057  0.146 0.440 1.003    3000
Sigma[2,2]    0.243 0.154  0.082  0.144  0.205  0.295 0.631 1.002    1100
mean.phi[1]   0.549 0.067  0.419  0.504  0.549  0.594 0.678 1.002    1400
mean.phi[2]   0.669 0.039  0.593  0.644  0.669  0.694 0.749 1.001    2200

```

$\Sigma_{11}$  (note that this is sigma[1,1] in the table above) is the temporal variance of the logit male survival, and  $\Sigma_{22}$  is that for logit female survival.

The elements  $\Sigma_{12}$  and  $\Sigma_{21}$  are the temporal covariances of logit male and logit female survival. This quantity may not be easy to interpret, and we may want to compute the temporal correlation of male and female survival:

```
corr.coef <- cjs.corr$sims.list$Sigma[,1,2] / sqrt(cjs.corr$sims.
  list$Sigma[,1,1] * cjs.corr$sims.list$Sigma[,2,2])
```

The mean and the credible interval of the correlation coefficient ( $\rho$ ) are 0.16 (-0.43, 0.67), and the probability that  $\rho > 0$  is 0.71. As usual these quantities are computed from the posterior distribution of `corr.coef`.

## 7.7 MODELS WITH AGE EFFECTS

Survival often changes with age. For most species, survival in their first year of life is lower than later. In addition, with senescence, survival may decline in older age classes. Therefore, we might want to estimate different survival parameters for each age class. To model age effects on survival, individuals must be aged when they are first captured, although recently developed models allow relaxing this assumption for some of the individuals (Pledger et al., 2009). We create a matrix  $x_{i,t}$  indicating the age at each time  $t$  for each individual  $i$ . For example, assume a study over 6 years and two individuals that are first captured at the second occasion. The elements of matrix  $x$  would then be [NA 1 2 3 4] for the first individual that was born at the second occasion and [NA 2 3 4 5] for the second individual that was 1-year old at the second occasion. We can then model survival as a function of age  $x$  as follows:

$$\text{logit}(\phi_{i,t}) = \mu + \beta x_{i,t} + \varepsilon_i \\ \varepsilon_i \sim \text{Normal}(0, \sigma^2).$$

This model can be adapted very flexibly. First, we may include an individual random effect ( $\varepsilon_i$ ) as already shown above. Inclusion of individual “frailty” can be important, if we aim at estimating senescence. Second, we may assume that survival changes linearly with age (as in the formula above), that it changes nonlinearly with age (e.g., Gaillard et al., 2004), or that  $x$  is a categorical variable. The last is perhaps the most frequent type of model for age effects adopted in practice. If we distinguish only two age classes (i.e., separate survival during the first year of life vs. later), the elements of matrix  $x$  would become [NA 1 2 2 2] for the first individual above and [NA 2 2 2 2] for the second individual. Finally, one might include time effects in addition to age effects. A general formation might then be

$$\text{logit}(\phi_{i,t}) = \beta_{x(i,t)} + \varepsilon_i \\ \varepsilon_i \sim \text{Normal}(0, \sigma^2),$$

where  $\beta_{x(i,t)}$  are the effects of age class  $x$  of individual  $i$  at time  $t$  and  $\varepsilon_i$  are individual frailty terms. Note that a principal difference between the first and the second model is that the age variable  $x$  is continuous in the first but categorical in the second model.

To illustrate the model, we consider a simple example, in which juvenile and adult little owls are marked. We assume that survival in the first year of life (from age 0 to age 1 year) is different from survival in subsequent age classes (from age 1 year onward). Thus, we need a model with two age classes for survival. We simulate data first, by creating two data sets, one for individuals marked as juveniles, and one for individuals marked as adults. We then construct matrix  $x$  for each age class and merge the two data sets and matrices ( $x$ ).

```
# Define parameter values
n.occasions <- 10 # Number of capture occasions
marked.j <- rep(200, n.occasions-1) # Annual number of newly marked
# juveniles
marked.a <- rep(30, n.occasions-1) # Annual number of newly marked
# adults
phi.juv <- 0.3 # Juvenile annual survival
phi.ad <- 0.65 # Adult annual survival
p <- rep(0.5, n.occasions-1) # Recapture
phi.j <- c(phi.juv, rep(phi.ad, n.occasions-2))
phi.a <- rep(phi.ad, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI.J <- matrix(0, ncol = n.occasions-1, nrow = sum(marked.j))
for (i in 1:length(marked.j)){
  PHI.J[(sum(marked.j[1:i])-marked.j[i]+1):sum(marked.j[1:i]),
  i:(n.occasions-1)] <- matrix(rep(phi.j[1:(n.occasions-i)],
  marked.j[i]), ncol = n.occasions-i, byrow = TRUE)
}
P.J <- matrix(rep(p, sum(marked.j)), ncol = n.occasions-1,
nrow = sum(marked.j), byrow = TRUE)
PHI.A <- matrix(rep(phi.a, sum(marked.a)), ncol = n.occasions-1,
nrow = sum(marked.a), byrow = TRUE)
P.A <- matrix(rep(p, sum(marked.a)), ncol = n.occasions-1,
nrow = sum(marked.a), byrow = TRUE)

# Apply simulation function
CH.J <- simul.cjs(PHI.J, P.J, marked.j)
CH.A <- simul.cjs(PHI.A, P.A, marked.a)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f.j <- apply(CH.J, 1, get.first)
f.a <- apply(CH.A, 1, get.first)

# Create matrices X indicating age classes
x.j <- matrix(NA, ncol = dim(CH.J)[2]-1, nrow = dim(CH.J)[1])
x.a <- matrix(NA, ncol = dim(CH.A)[2]-1, nrow = dim(CH.A)[1])
for (i in 1:dim(CH.J)[1]) {
```

```

for (t in f.j[i]:(dim(CH.J)[2]-1)) {
  x.j[i,t] <- 2
  x.j[i,f.j[i]] <- 1
} #t
} #i
for (i in 1:dim(CH.A)[1]) {
  for (t in f.a[i]:(dim(CH.A)[2]-1)) {
    x.a[i,t] <- 2
  } #t
} #i

```

Next, we combine the two data sets into a common set.

```

CH <- rbind(CH.J, CH.A)
f <- c(f.j, f.a)
x <- rbind(x.j, x.a)

```

Finally, we define the model in BUGS language and fit it to the data. We treat age as a categorical variable, so we use the identity link.

```

# Specify model in BUGS language
sink("cjs-age.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)) {
    phi[i,t] <- beta[x[i,t]]
    p[i,t] <- mean.p
  } #t
} #i
for (u in 1:2) {
  beta[u] ~ dunif(0, 1)           # Priors for age-specific survival
}
mean.p ~ dunif(0, 1)                 # Prior for mean recapture

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions) {
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
}

", fill = TRUE)
sink()

```

```

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH) [1], n.occasions =
dim(CH) [2], z = known.state.cjs(CH), x = x)

# Initial values
inits <- function() {list(z = cjs.init.z(CH, f), beta = runif(2, 0, 1),
mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("beta", "mean.p")

# MCMC settings
ni <- 2000
nt <- 3
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
cjs.age <- bugs(bugs.data, inits, parameters, "cjs-age.bug", n.chains =
nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
bugs.directory = bugs.dir, working.directory = getwd())

```

The model runs slowly, but convergence is achieved after only 1000 samples. The parameter estimates are close to the parameters used for the simulations.

```

print(cjs.age, digits = 3)

      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
beta[1] 0.317 0.015 0.287 0.306 0.318 0.328 0.347 1.002    880
beta[2] 0.666 0.015 0.638 0.657 0.667 0.676 0.695 1.001   1000
mean.p  0.486 0.019 0.452 0.473 0.486 0.499 0.525 1.005    410

```

It is straightforward to include other models for the age effect. Depending on the models that we want to fit, matrix  $x$  needs to be adapted. If we want to model survival as a linear function of age,  $x$  must indicate the true age in each year. If the goal is to treat age as a categorical variable,  $x$  must include as many categories as we want to distinguish (e.g., two above). Then the GLM, which relates survival to  $x$ , needs to be adapted. For example, if survival is modeled as a linear function of age, we first create  $x$  and only include into the analysis individuals marked as juveniles.

```

# Create matrix X indicating age classes
x <- matrix(NA, ncol = dim(CH) [2]-1, nrow = dim(CH) [1])
for (i in 1:dim(CH) [1]) {
  for (t in f[i]:(dim(CH) [2]-1)) {
    x[i,t] <- t-f[i]+1
  } #t
} #i

```

As usual, the BUGS model needs a few changes in the “Priors and constraints” part:

```
# Priors and constraints
for (i in 1:nind) {
  for (t in f[i] : (n.occasions-1)) {
    logit(phi[i,t]) <- mu + beta*x[i,t]
    p[i,t] <- mean.p
  } #t
} #i
mu ~ dnorm(0, 0.01)           # Prior for mean of logit survival
beta ~ dnorm(0, 0.01)          # Prior for slope parameter
for (i in 1:(n.occasions-1)) {
  phi.age[i] <- 1 / (1+exp(-mu -beta*i))  # Logit back-transformation
}
mean.p ~ dunif(0, 1)           # Prior for mean recapture
```

Age effects can also be combined with time effects in a very similar way as we have seen with group effects (see Section 7.6). Models can be specified in which survival of the defined age classes vary independently from each other across time, in which the temporal pattern of the age classes is additive, or in which only survival of one age class is time-dependent. Models with random time effects are also useful, allowing the temporal variability of survival of each age class to be modeled independently, or in which the temporal correlation is estimated. It is also possible to consider cohort effects (Reid et al., 2003), that is, the survival of individuals born in one cohort (year) is different from the survival of individuals born in another cohort. This requires that we define a variable indicating the cohort for each individual. In fact, for individuals that are young when marked, our vector  $f$  already includes this information. Survival is then modeled as a function of  $f$ , we may consider it to be fixed or random, and we can combine it with additional time and/or age effects. Care must be taken with model specification because a model with cohort  $\times$  time interaction is the same as a model with age  $\times$  time interaction or one with cohort  $\times$  age interaction.

## 7.8 IMMEDIATE TRAP RESPONSE IN RECAPTURE PROBABILITY

---

One assumption of standard capture–recapture models is that all marked animals alive and available for capture at a given occasion have the same capture probability. Sometimes, this assumption is violated in a very specific way, namely when individuals captured at time  $t - 1$  have a different recapture probability at time  $t$  than individuals not captured at time  $t - 1$ . This is called immediate trap response (see also Section 6.2.3).

If recapture probability at time  $t$  for individuals captured at  $t - 1$  is higher than for individuals not captured at  $t - 1$ , this is “trap-happiness” and if recapture probability is lower, then it is called “trap-shyness”. Trap-happiness can occur if baited traps are used, and trap-shyness can occur if the interval between capture occasions is short (Pradel, 1993). These effects may also be induced by the sampling method and not reflect a behavioral change of the individuals. However, trap response must be modeled; otherwise, survival estimates will be biased. To account for immediate trap response, a multistate model can be used (Gimenez et al., 2003; Schaub et al., 2009; Appendix 2.2), but here we will use a single-state model and model recapture as a function of whether or not an individual was captured at the preceding occasion. We need, therefore, to construct a matrix  $m$  that contains this information. The element of  $m$  for individual  $i$  at time  $t$  takes value 1 if individual  $i$  was captured at  $t - 1$ , and value 2 otherwise. The recapture probability is then modeled as

$$p_{i,t} = \beta_{m(i,t)},$$

where  $\beta_m$  takes two values, depending on whether  $m_{i,t}$  is 1 or 2. We may also include additive time effects and use the logit link function,

$$\text{logit}(p_{i,t}) = \beta_{m(i,t)} + \gamma_t.$$

The model with interaction between time and behavioral response is parameter-redundant (Gimenez et al., 2003).

Simulating such data is best done with a multistate model (see Appendix 2.2). For illustration, we imagine that we wish to estimate survival of red-backed shrikes (Fig. 7.7), a beautiful bird species of hedgerows. We catch adults during the breeding season, mark them with color rings to facilitate resighting in subsequent years, and survey all potential breeding territories each year. Typically, we focus on breeding territories that were occupied in previous years. If time allows, we search for other, newly established territories. Thus, marked individuals that survive and return to their territory have a higher chance of being resighted, while individuals that establish new territories are less likely to be found. However, once they are found, their chances of being resighted in the next year increase. Such a sampling protocol, which is not uncommon in studies of color-marked birds, induces a “trap-happy effect” which biases survival unless accounted for. For data simulation, we assume survival  $\phi = 0.55$  and resighting probabilities  $p_{ss} = 0.75$  following a sighting in the preceding year and  $p_{ns} = 0.35$  otherwise.

```
# Import data
CH <- as.matrix(read.table(file = "trap.txt", sep = " "))

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)
```



**FIGURE 7.7** Male red-backed shrike (*Lanius collurio*) feeding a fledgling (Photograph by D. Studler).

```
# Create matrix m indicating when an individual was captured
m <- CH[,1:(dim(CH)[2]-1)]
u <- which(m==0)
m[u] <- 2
```

The capture-histories of the first four individuals are as follows:

```
1 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 1 0 0 1 1 1 0
1 1 0 1 1 0 1 0
```

and the corresponding matrix  $m$  for these individuals is

```
1 1 2 2 2 2 2 2
1 2 2 2 2 2 2 2
1 1 2 2 1 1 1 1
1 1 2 1 1 2 1 1
```

Here a 1 denotes that an individual was captured at the preceding occasion, and a 2 denotes that it was not captured at the preceding occasion. Matrix  $m$  has as many columns as there are recapture parameters, thus one fewer than the total number of capture occasions.

The BUGS code to fit the trap-response model is as follows:

```
# Specify model in BUGS language
sink("cjs-trap.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)) {
    phi[i,t] <- mean.phi
    p[i,t] <- beta[m[i,t]]
  } #t
} #i
mean.phi ~ dunif(0, 1)           # Prior for mean survival
for (u in 1:2) {
  beta[u] ~ dunif(0, 1)           # Priors for recapture
}

# Likelihood components
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions) {
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
  } #t
} #i
},fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = CH, f = f, nind = dim(CH)[1], n.occasions =
  dim(CH)[2], z = known.state.cjs(CH), m = m)

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), mean.phi = runif(1, 0,
  1), beta = runif(2, 0, 1))}

# Parameters monitored
parameters <- c("mean.phi", "beta")

# MCMC settings
ni <- 20000
nt <- 3
nb <- 10000
nc <- 3
```

```
# Call WinBUGS from R (BRT 1 min)
cjs.trap <- bugs(bugs.data, inits, parameters, "cjs-trap.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())
```

The estimated parameters are close to the parameters used for the simulation.

```
print(cjs.trap, digits = 3)

    mean      sd  2.5%   25%   50%   75% 97.5%   Rhat n.eff
mean.phi 0.567  0.076 0.462 0.515 0.552 0.602 0.763 1.006  2400
beta[1]   0.756  0.091 0.547 0.701 0.770 0.823 0.897 1.006  2700
beta[2]   0.379  0.207 0.063 0.210 0.359 0.527 0.814 1.003  4500
```

This approach is again very flexible and can be extended easily. For example, if an individual may be captured more than once during an occasion, those captured more may have a higher capture probability. By including the information about how many times an individual was captured, we can adjust for this sort of capture heterogeneity (Fletcher, 1994). The matrix  $m$  then contains the number of times an individual is caught at an occasion, and recapture is modeled as a function of  $m$ .

## 7.9 PARAMETER IDENTIFIABILITY

In principle, we are quite free to specify any among a large number of models, especially when using BUGS. However, there is no guarantee that all parameters in a fitted model are indeed identified, that is, can be estimated. In fact, it is common that some parameters are not identifiable. There are two kinds of nonidentifiability: intrinsic and extrinsic. A model has intrinsically identifiable parameters if the same likelihood for the data cannot be obtained by a smaller number of parameters, while parameter-redundant models (those with at least one unidentified parameter) can be expressed in terms of fewer than the original number of parameters (Catchpole and Morgan, 1997). Extrinsic nonidentifiability refers to the situation where a parameter should be identifiable given the structure of a model but is not because the particular data set is insufficient in some regard. Thus, intrinsic nonidentifiability is a feature of a model while extrinsic nonidentifiability is a feature of a data set. Intrinsic nonidentifiability of models can be studied without data using symbolic algebra (Catchpole and Morgan, 1997; Catchpole et al., 2001; Gimenez et al., 2003, 2004) or the analysis of “perfect” data (analytic-numeric method; Burnham et al., 1987), while extrinsic nonidentifiability is best studied using simulation (e.g., Schaub et al., 2004a; Schaub, 2009; Bailey et al., 2010). Of course, intrinsic and extrinsic nonidentifiability may occur together for a particular model and data set.

In the Bayesian framework, the topic of nonidentifiability is slightly different. Because the posterior is a combination of the likelihood and the prior, the posterior is defined (provided that the prior is proper; Gelman et al., 2004). However, if the information in the data is very low for a particular parameter (i.e., there is extrinsic nonidentifiability) or if the likelihood surface is completely flat for a parameter (intrinsic nonidentifiability), then the posterior will simply reflect the prior for that parameter. Therefore, a prior sensitivity analysis can give insights into the identifiability of a parameter. Gimenez et al. (2009b) developed an approach to assess parameter identifiability based on this idea. Using flat priors for parameters, they compared the overlap between the prior and the posterior. If the overlap between the two distributions is large, a parameter is weakly identifiable.

Here, we illustrate this with a well-known example. In the classical, fully time-dependent CJS model  $\{\phi_t, p_t\}$ , the last survival and the last recapture probability are not identifiable—it is only possible to estimate the product of the two (Lebreton et al., 1992). Thus, this is an intrinsic identifiability problem. In the following example, we fit the model  $\{\phi_t, p_t\}$  to the data and use flat priors for all parameters. We then inspect the posterior and the prior of some survival and recapture parameters.

```
# Define parameter values
n.occasions <- 12 # Number of capture occasions
marked <- rep(30, n.occasions-1) # Annual number of newly marked
# individuals
phi <- c(0.6, 0.5, 0.55, 0.6, 0.5, 0.4, 0.6, 0.5, 0.55, 0.6, 0.7)
p <- c(0.4, 0.65, 0.4, 0.45, 0.55, 0.68, 0.66, 0.28, 0.55, 0.45, 0.35)

# Define matrices with survival and recapture probabilities
PHI <- matrix(phi, ncol = n.occasions-1, nrow = sum(marked), byrow = TRUE)
P <- matrix(p, ncol = n.occasions-1, nrow = sum(marked), byrow = TRUE)

# Simulate capture-histories
CH <- simul.cjs(PHI, P, marked)

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Specify model in BUGS language
sink("cjs-t-t.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
    for (t in f[i]:(n.occasions-1)) {
        phi[i,t] <- phi.t[t]
        p[i,t] <- p.t[t]
    } #
} #i
}
```

```

for (t in 1:(n.occasions-1)){
  phi.t[t] ~ dunif(0, 1)          # Priors for time-spec. survival
  p.t[t] ~ dunif(0, 1)           # Priors for time-spec. recapture
}

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- 1
  for (t in (f[i]+1):n.occasions){
    # State process
    z[i,t] ~ dbern(mu1[i,t])
    mu1[i,t] <- phi[i,t-1] * z[i,t-1]
    # Observation process
    y[i,t] ~ dbern(mu2[i,t])
    mu2[i,t] <- p[i,t-1] * z[i,t]
    } #t
  } #i
}
",fill = TRUE)
sink()

# Bundle data
bugs.data<- list(y = CH, f = f, nind = dim(CH) [1], n.occasions = dim(CH) [2],
z = known.state.cjs(CH))

# Initial values
inits <- function(){list(z = cjs.init.z(CH, f), phi.t = runif((dim(CH)
[2]-1), 0, 1), p.t = runif((dim(CH) [2]-1), 0, 1))}

# Parameters monitored
parameters <- c("phi.t", "p.t")

# MCMC settings
ni <- 25000
nt <- 3
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT 7 min)
cjs.t.t <- bugs(bugs.data, inits, parameters, "cjs-t-t.bug", n.chains =
nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
bugs.directory = bugs.dir, working.directory = getwd())

# Plot posterior distributions of some phi and p
par(mfrow = c(2, 2), cex = 1.2, las = 1, mar=c(5, 4, 2, 1))
plot(density(cjs.t.t$sims.list$phi.t[,6]), xlim = c(0, 1), ylim = c(0, 5),
main = "", xlab = expression(phi[6]), ylab = "Density", frame = FALSE,
lwd = 2)
abline(h = 1, lty = 2, lwd = 2)
par(mar=c(5, 3, 2, 2))
plot(density(cjs.t.t$sims.list$phi.t[,11]), xlim = c(0, 1),
ylim = c(0, 5), main = "", xlab = expression(phi[11]), ylab = "",
frame = FALSE, lwd = 2)
abline(h = 1, lty = 2, lwd = 2)
par(mar=c(5, 4, 2, 1))

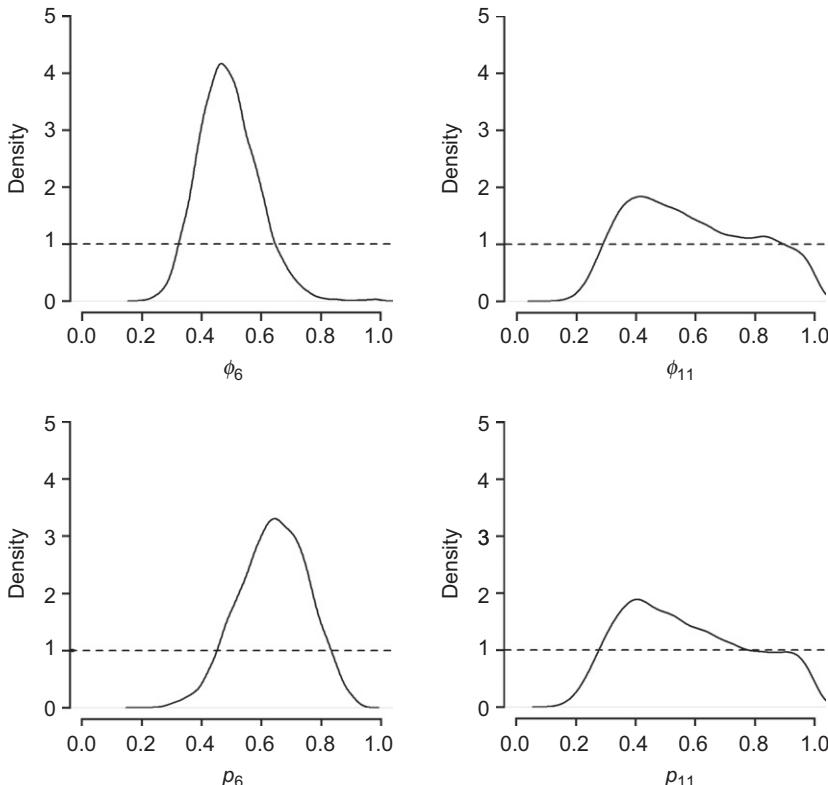
```

```

plot(density(cjs.t.t$sims.list$p.t[,6]), xlim = c(0, 1), ylim = c(0, 5),
      main = "", xlab = expression(p[6]), ylab = "Density", frame = FALSE,
      lwd = 2)
abline(h = 1, lty = 2, lwd = 2)
par(mar=c(5, 3, 2, 2))
plot(density(cjs.t.t$sims.list$p.t[,11]), xlim = c(0, 1), ylim =
      c(0, 5), main = "", xlab = expression(p[11]), ylab = "", frame = FALSE,
      lwd = 2)
abline(h = 1, lty = 2, lwd = 2)

```

To inspect the result, we plot the posterior and prior densities of some parameters (Fig. 7.8). It is obvious that  $\phi_6$  and  $p_6$  are identifiable: their posterior is nicely peaked and does not overlap much with the prior distribution. By contrast, the posterior distributions of  $\phi_{11}$  and  $p_{11}$  do not have a clear peak and the overlap with the prior is large. These



**FIGURE 7.8** Posterior density plots of the sixth and the last survival and recapture probabilities. The dotted line shows the prior density. The last parameters ( $\phi_{11}$  and  $p_{11}$ ) are not separately identifiable.

parameters are not identifiable. Gimenez et al. (2009b) developed a quantitative guideline based on the degree of overlap between posterior and prior to decide when a parameter is identifiable. Note that when the year effects are not fixed as above, but random, the problem of nonidentifiability disappears because information from survival and recapture from the complete data set is used to estimate the last parameters. You may want to try this!

In any analysis of capture–recapture models (or actually, of *any* model), you should be aware that some parameters might not be estimable, although WinBUGS (or another software) may give you estimates for all parameters (Lunn et al., 2010). Obviously, no inference can be made about nonidentifiable parameters. This challenge is even greater for multistate capture–recapture models (see Chapter 9).

## **7.10 FITTING THE CJS TO DATA IN THE M-ARRAY FORMAT: THE MULTINOMIAL LIKELIHOOD**

### **7.10.1 Introduction**

So far we have analyzed the individual capture-histories using a state-space formulation of the CJS model. This is a very general framework within which a multitude of different kinds of models can be formulated, but it comes at a computational cost. As all capture-histories are analyzed individually, a loop over all individuals is necessary. In addition, every unknown latent state (e.g., individual survival) needs to be estimated. Capture–recapture data can, however, also be summarized in the so-called m-array (Burnham et al., 1987). The CJS model is then fitted using a multinomial likelihood. This has the advantage of much faster computation, but the disadvantage of reduced flexibility in the modeling. In particular, models with individual effects can no longer be fitted.

We first introduce the m-array format, by considering the following example—we have capture-histories of seven individuals:

```
1 0 1 0
1 1 0 0
1 1 0 0
1 0 0 0
0 1 1 1
0 1 0 0
0 0 1 0
```

The m-array tabulates the number of individuals released at one occasion that are next recaptured on each subsequent occasion. It is a triangular matrix, in which rows refer to release occasions and columns refer

to recapture occasions. An additional column tallies up the individuals that are not recaptured. To create the m-array, the capture-histories of all individuals are broken into fragments. The number of captures equals the number of fragments. Each fragment considers the last release occasion and the next recapture occasion. For example, the capture-history [1 0 1 0] is broken into the two fragments [1 0 1 0] and [0 0 1 0]. The first fragment shows that the individual was released at occasion 1 and first recaptured at occasion 3. The second fragment shows that the individual was released on occasion 3 and was never recaptured. The m-array for the seven capture-histories above is:

Release Occasion	Recapture Occasion				Never Recaptured
	2	3	4		
1	2	1	0		1
2	-	1	0		3
3	-	-	1		2

Fitting the CJS model to the data using the m-array implicitly assumes the absence of any individual effects on survival and recapture probabilities. By summarizing the data in this form, it is evident that effects of individual covariates cannot be fitted because the capture-histories of the individuals are broken up. Age as a special class of individual covariate can be considered but requires a different format of the m-array (see Section 7.10.3). Otherwise, all the information that originally was included in the individual capture-histories is kept; it is just summarized in the form of minimal sufficient statistics.

The following R function converts capture-histories into the m-array format.

```
# Function to create a m-array based on capture-histories (CH)
marray <- function(CH) {
  nind <- dim(CH)[1]
  n.occasions <- dim(CH)[2]
  m.array <- matrix(data = 0, ncol = n.occasions+1, nrow =
  n.occasions)

  # Calculate the number of released individuals at each time period
  for (t in 1:n.occasions) {
    m.array[t,1] <- sum(CH[,t])
  }
  for (i in 1:nind) {
    pos <- which(CH[i,] != 0)
    g <- length(pos)
```

```

for (z in 1:(g-1)) {
  m.array[pos[z],pos[z+1]] <- m.array[pos[z],pos[z+1]] + 1
} #z
} #i

# Calculate the number of individuals that is never recaptured
for (t in 1:n.occasions) {
  m.array[t,n.occasions+1] <- m.array[t,1] -
    sum(m.array[t,2:n.occasions])
}
out <- m.array[1:(n.occasions-1),2:(n.occasions+1)]
return(out)
}

```

The expected values of the entries of the m-array are given based on the underlying model parameters ( $\phi_t$  and  $p_t$ ) and the number of released individuals. These define the cell probabilities of the multinomial distributions for each release occasion.

Release Occasion	Recaptured at Occasion			Never Recaptured
	2	3	4	
1	$\phi_1 p_1$	$\phi_1(1 - p_1)$	$\phi_1(1 - p_1)$	$1 - \phi_1 p_1 - \phi_1(1 - p_1)\phi_2 p_2 - \phi_1(1 - p_1)$
		$\phi_2 p_2$	$\phi_2(1 - p_2)$	$\phi_2(1 - p_2)\phi_3 p_3 = 1 - \Sigma(\text{rel. occ 1})$
			$\phi_3 p_3$	
2	0	$\phi_2 p_2$	$\phi_2(1 - p_2)$	$1 - \phi_2 p_2 - \phi_2(1 - p_2)\phi_3 p_3 = 1 - \Sigma(\text{rel. occ 2})$
3	0	0	$\phi_3 p_3$	$1 - \phi_3 p_3 = 1 - \Sigma(\text{rel. occ 3})$

Note: The entry in cell (1,3) is the product  $\phi_1(1 - p_1)\phi_2 p_2$  (and likewise for the other cells).

### 7.10.2 Time-Dependent Models

The rows of the observed m-array data follow a multinomial distribution with index equal to the number of released individuals at each occasion and the cell probabilities that are functions of survival and recapture parameters, as shown in the table above. Fitting this model in BUGS is straightforward: essentially, we only need to define the cell probabilities of the m-array.

Using the m-array formulation of the CJS model, it is also quite easy to assess the fit of the model, that is, to compute a Bayesian  $p$ -value based on the posterior predictive distribution of a goodness-of-fit (GOF) statistic (see Gelman et al., 1996, 2004; Section 12.3). This technique for GOF assessment is also called posterior predictive checking because its rationale is based on a comparison of data simulated (predicted) under the model, and the actual data set that is analyzed using that

model. Simulated data sets under a model are obtained easily as part of the MCMC updating from the posterior predictive distribution of the data. Usually, some discrepancy measure is calculated that measures how “far apart” the data are from their expected values under the model. Often, omnibus test statistics such as chi-squared are used as a discrepancy measure, but other statistics may be chosen to specifically highlight how well a model fits the data in some particular manner, for instance, how well it describes extreme values (Gelman et al., 1996). This discrepancy measure is calculated for both the simulated and the actual data set. The values of both discrepancy measures change at each iteration of the MCMC simulation algorithm because the parameter values change with each iteration as well and they are used both to generate a replicate data set and to compute the expected values for the data. At the end of the posterior sampling, one has as many draws from the posterior distribution of the chosen discrepancy measure for the simulated (perfect) data sets as for the actual data sets. The simulated data sets are “perfect” in the sense that they were generated under exactly the same model that is used for parameter estimation in the observed data and using the exact parameter values obtained in that analysis. The posterior draws of the discrepancy measure for the replicate data, therefore, provide the reference distribution for the discrepancy measure under the null hypothesis that the model fits our data. The proportion of times that the discrepancy measure for the simulated data sets is more extreme than that for the actual data set is called a Bayesian  $p$ -value. Under the null hypothesis that the model in question is the data-generating model, this should happen about 50% of times; hence, Bayesian  $p$ -values close to 0 or 1 are suspicious. A graph of the values of the discrepancy measure from the replicate data sets plotted against those for the actual data sets may be even more informative than the scalar Bayesian  $p$ -value to point out ways how a model may not fit. The value of the  $p$ -value represents the proportion of points that lie above the 1:1 line of equality.

Bayesian  $p$ -values have been criticized for several reasons. First, they use the data twice (once, to generate replicate data sets and then to compute the expected data and compare that with both the replicate and the actual data sets). They may thus not be strict enough and not reject often enough the hypothesis of a fitting model. Second, it is unclear what value of a Bayesian  $p$ -value represents a good fit. For instance, there would be no objective way of saying that values outside of the interval (0.05, 0.95) represent models that do not fit the data. Thus, Bayesian  $p$ -values are a descriptive technique only. And finally, the rationale underlying a Bayesian  $p$ -value is intrinsically frequentist: Learning from the data is not limited to the information content in the actual data set but instead based on hypothetical replicate data sets as well. This may be offensive to hardcore Bayesians who adhere to the so-called likelihood principle (Lindley, 2006),

which says that all information about a data set is contained in the likelihood function. Our own position in this respect is pragmatic: We like Bayesian  $p$ -values as a simple and very flexible way of pointing out ways in which a model may not fit a data set.

In the current example of a survival analysis, we could not test the GOF of a state-space model for binary responses (observed vs. not observed). The reason for this is that discrepancy measures such as the deviance are uninformative about model fit for binary responses (McCullagh and Nelder, 1989). GOF can, however, be assessed for some summary of binary responses and the m-array represents just one such summary. So here now, we create replicate data (i.e., m-arrays,  $e_{ij}$ ), and compare the observed ( $x_{ij}$ ) and the expected m-arrays using a discrepancy measure. We could use the  $\chi^2$ -discrepancy as in Chapter 12, but instead follow Brooks et al. (2000b) and use the Freeman-Tukey statistic ( $D = \sum(x_{ij}^{1/2} - e_{ij}^{1/2})^2$ ). It makes unnecessary to pool cells with small expected values. The Freeman-Tukey statistic is computed for the observed and simulated data.

We use the data as created in Section 7.9 to illustrate the use of the model. The following code fits the CJS model using the multinomial likelihood and includes the posterior predictive check.

```
# Specify model in BUGS language
sink("cjs-mnl.bug")
cat("
model {

# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi[t] ~ dunif(0, 1)           # Priors for survival
  p[t] ~ dunif(0, 1)             # Priors for recapture
}

# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
  marr[t,1:n.occasions] ~ dmulti(pr[t, ], r[t])
}

# Calculate the number of birds released each year
for (t in 1:(n.occasions-1)){
  r[t] <- sum(marr[t, ])
}

# Define the cell probabilities of the m-array
# Main diagonal
for (t in 1:(n.occasions-1)){
  q[t] <- 1-p[t]                 # Probability of non-recapture
  pr[t,t] <- phi[t]*p[t]
  # Above main diagonal
  for (j in (t+1):(n.occasions-1)){
    pr[t,j] <- prod(phi[t:j])*prod(q[t:(j-1)])*p[j]
    } #j
}
```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr[t,j] <- 0
} #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr[t,n.occasions] <- 1-sum(pr[t,1:(n.occasions-1)])
} #t

# Assess model fit using Freeman-Tukey statistic
# Compute fit statistics for observed data
for (t in 1:(n.occasions-1)){
  for (j in 1:n.occasions){
    expmarr[t,j] <- r[t]*pr[t,j]
    E.org[t,j] <- pow((pow(marr[t,j], 0.5)-pow(expmarr[t,j],
      0.5)), 2)
  } #j
} #t

# Generate replicate data and compute fit stats from them
for (t in 1:(n.occasions-1)){
  marr.new[t,1:n.occasions] ~ dmulti(pr[t, ], r[t])
  for (j in 1:n.occasions){
    E.new[t,j] <- pow((pow(marr.new[t,j], 0.5)-pow(expmarr[t,j],
      0.5)), 2)
  } #j
} #t
fit <- sum(E.org[,])
fit.new <- sum(E.new[,])
}
",fill = TRUE)
sink()

# Create the m-array from the capture-histories
marr <- marray(CH)

# Bundle data
bugs.data <- list(marr = marr, n.occasions = dim(marr) [2])

# Initial values
inits <- function() {list(phi = runif(dim(marr) [2]-1, 0, 1),
  p = runif(dim(marr) [2]-1, 0, 1))}

# Parameters monitored
parameters <- c("phi", "p", "fit", "fit.new")

# MCMC settings
ni <- 10000
nt <- 3
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
cjs <- bugs(bugs.data, inits, parameters, "cjs-mnl.bug", n.chains =
  nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

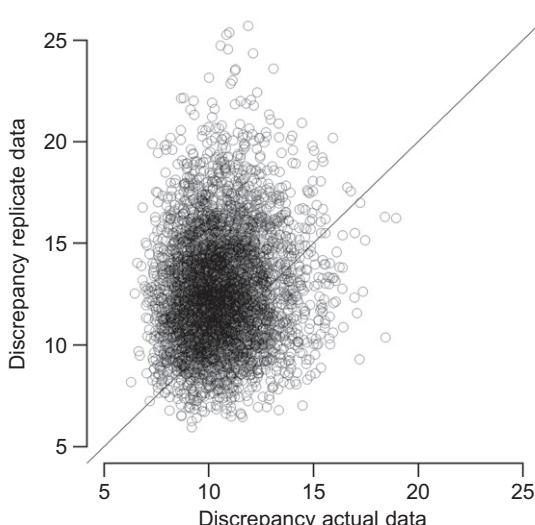
```

```
print(cjs, digits = 3)

      mean     sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
phi[1] 0.632 0.167 0.331  0.505  0.621  0.754 0.960 1.001 5000
[...]
p[11]  0.577 0.206 0.261  0.406  0.547  0.742 0.968 1.006  380
fit    10.563 1.674 7.773  9.378 10.400 11.570 14.320 1.001 5000
fit.new 12.671 2.744 8.095 10.720 12.420 14.330 18.830 1.002 2400
```

The model converges quickly and the MCMC samples are obtained in a short time (to compare, you may use the same data and run the corresponding state-space model of Section 7.3). The comparison of the discrepancy between the observed and the simulated data (Fig. 7.9) shows that they are similar, suggesting that the model is adequate for the data set. This is confirmed by a Bayesian  $p$ -value of 0.75. For more discussion about checking of capture–recapture models, see Brooks et al. (2000a, 2000b) and King et al. (2010).

```
# Evaluation of fit
plot(cjs$sims.list$fit, cjs$sims.list$fit.new, xlab = "Discrepancy
actual data", ylab = "Discrepancy replicate data", las = 1,
ylim = c(5, 25), xlim = c(5, 25), bty = "n")
abline(0, 1, col = "black", lwd = 2)
mean(cjs$sims.list$fit.new > cjs$sims.list$fit)
```



**FIGURE 7.9** Posterior predictive check of model fit by a scatter plot of the discrepancy measure for replicate (simulated) versus actual (observed) data in a CJS model. The Bayesian  $p$ -value is the proportion of points above the 1:1 line.

Construction of models with time effects (fixed or random) using the multinomial likelihood requires changes in the “Priors and constraints” part of the model code, in exactly the same way as we have introduced for the state-space formulation. However, the formulation of models with groups is slightly different. We need to create an m-array for each group and to write a separate likelihood (with different parameters) for each data set. Once this is done, we can constrain the group-specific parameters in the same way as with models using the state-space formulation (i.e., we can regard groups as fixed or as random, and we can combine them with time effects). See exercise 2 in [Section 7.13](#) for an example.

### 7.10.3 Age-Dependent Models

Models with age-dependent survival fitted with the multinomial likelihood need some adaptations (m-array, analyzing code), and we show this in detail using an example. We look at the situation in which young and adult little owls are marked and assume that survival in the first year of life (from age 0 to age 1 year) is different from survival in subsequent age classes (from age 1 onward). We first start with the simulation of the data. We will create two data sets, one for individuals marked as juveniles, and another for individuals marked as adults.

```
# Define parameter values
n.occasions <- 12 # Number of capture occasions
marked.j <- rep(200, n.occasions-1) # Annual number of newly marked juveniles
marked.a <- rep(30, n.occasions-1) # Annual number of newly marked adults
phi.juv <- 0.3 # Juvenile annual survival
phi.ad <- 0.65 # Adult annual survival
p <- rep(0.5, n.occasions-1) # Recapture
phi.j <- c(phi.juv, rep(phi.ad, n.occasions-2))
phi.a <- rep(phi.ad, n.occasions-1)

# Define matrices with survival and recapture probabilities
PHI.J <- matrix(0, ncol = n.occasions-1, nrow = sum(marked.j))
for (i in 1:(length(marked.j)-1)){
  PHI.J[(sum(marked.j[1:i])-marked.j[i]+1):sum(marked.j[1:i]), i:(n.occasions-1)] <-
    matrix(rep(phi.j[1:(n.occasions-i)], marked.j[i]),
    ncol = n.occasions-i, byrow = TRUE)
}
P.J <- matrix(rep(p, n.occasions*sum(marked.j)), ncol =
  n.occasions-1, nrow = sum(marked.j), byrow = TRUE)
PHI.A <- matrix(rep(phi.a, sum(marked.a)), ncol = n.occasions-1,
  nrow = sum(marked.a), byrow = TRUE)
P.A <- matrix(rep(p, sum(marked.a)), ncol = n.occasions-1,
  nrow = sum(marked.a), byrow = TRUE)
```

```
# Apply simulation function
CH.J <- simul.cjs(PHI.J, P.J, marked.j)
CH.A <- simul.cjs(PHI.A, P.A, marked.a)
```

Next, we create two m-arrays, one for juveniles and another for adults. The difficulty is that whenever an individual initially marked as a juvenile is recaptured, it has become an adult. Thus, it must be “released” in the m-array of the individuals initially marked as adults. To achieve this goal, we first split the capture-histories of individuals marked as juveniles based on whether or not they were ever recaptured (recaptured at least once: CH.J.R, never recaptured: CH.J.N). The first capture of CH.J.R is then removed, the resulting capture-histories added to the capture-histories of the individuals marked as adults and the m-array computed. Next, all recaptures after the first recapture of the original CH.J.R matrix are removed and the m-array computed. Because all these individuals are released as adults, the last columns of the m-array summarizing the number of individuals never recaptured have to be set to zero. Finally, we create the m-array for CH.J.N and add it to the previous m-array. The following code performs these data manipulations.

```
cap <- apply(CH.J, 1, sum)
ind <- which(cap >= 2)
CH.J.R <- CH.J[ind,]      # Juvenile CH recaptured at least once
CH.J.N <- CH.J[-ind,]     # Juvenile CH never recaptured

# Remove first capture
first <- numeric()
for (i in 1:dim(CH.J.R) [1]){
  first[i] <- min(which(CH.J.R[i,]==1))
}
CH.J.R1 <- CH.J.R
for (i in 1:dim(CH.J.R) [1]){
  CH.J.R1[i,first[i]] <- 0
}

# Add grown-up juveniles to adults and create m-array
CH.A.m <- rbind(CH.A, CH.J.R1)
CH.A.marray <- marray(CH.A.m)

# Create CH matrix for juveniles, ignoring subsequent recaptures
second <- numeric()
for (i in 1:dim(CH.J.R1) [1]){
  second[i] <- min(which(CH.J.R1[i,]==1))
}
CH.J.R2 <- matrix(0, nrow = dim(CH.J.R) [1], ncol = dim(CH.J.R) [2])
for (i in 1:dim(CH.J.R) [1]){
  CH.J.R2[i,first[i]] <- 1
  CH.J.R2[i,second[i]] <- 1
}

# Create m-array for these
CH.J.R.marray <- marray(CH.J.R2)
```

```
# The last column ought to show the number of juveniles not recaptured
# again and should all be zeros, since all of them are released as adults
CH.J.R.marray[,dim(CH.J) [2]] <- 0

# Create the m-array for juveniles never recaptured and add it to the
# previous m-array
CH.J.N.marray <- marray(CH.J.N)
CH.J.marray <- CH.J.R.marray + CH.J.N.marray
```

Now we write the BUGS code for the age-dependent model. We specify two component likelihoods, one for the m-array of adults and another for the m-array of juveniles. The code for adults is exactly the same as before (Section 7.10.2), but the code for juveniles has some twists. Here, the first survival for each release cohort (juvenile survival) is different from subsequent survival (which is that of adults).

```
# Specify model in BUGS language
sink("cjs-mnl-age.bug")
cat("
model {

# Priors and constraints
for (t in 1:(n.occasions-1)){
    phi.juv[t] <- mean.phi.juv
    phi.ad[t] <- mean.phi.ad
    p[t] <- mean.p
}
mean.phi.juv ~ dunif(0, 1)           # Prior for mean juv. survival
mean.phi.ad ~ dunif(0, 1)           # Prior for mean ad. survival
mean.p ~ dunif(0, 1)                # Prior for mean recapture

# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
    marr.j[t,1:n.occasions] ~ dmulti(pr.j[t,], r.j[t])
    marr.a[t,1:n.occasions] ~ dmulti(pr.a[t,], r.a[t])
}

# Calculate the number of birds released each year
for (t in 1:(n.occasions-1)){
    r.j[t] <- sum(marr.j[t,])
    r.a[t] <- sum(marr.a[t,])
}

# Define the cell probabilities of the m-arrays
# Main diagonal
for (t in 1:(n.occasions-1)){
    q[t] <- 1-p[t]                  # Probability of non-recapture
    pr.j[t,t] <- phi.juv[t]*p[t]
    pr.a[t,t] <- phi.ad[t]*p[t]
    # Above main diagonal
    for (j in (t+1):(n.occasions-1)){
        pr.j[t,j] <- phi.juv[t]*prod(phi.ad[(t+1):j])*prod(q[t:(j-1)])*p[j]
        pr.a[t,j] <- prod(phi.ad[t:j])*prod(q[t:(j-1)])*p[j]
    } #j
}
```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr.j[t,j] <- 0
  pr.a[t,j] <- 0
} #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr.j[t,n.occasions] <- 1-sum(pr.j[t,1:(n.occasions-1)])
  pr.a[t,n.occasions] <- 1-sum(pr.a[t,1:(n.occasions-1)])
} #t
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(marr.j = CH.J.marray, marr.a = CH.A.marray,
  n.occasions = dim(CH.J.marray) [2])

# Initial values
inits <- function(){list(mean.phijuv = runif(1, 0, 1), mean.phiad =
  runif(1, 0, 1), mean.p = runif(1, 0, 1))}

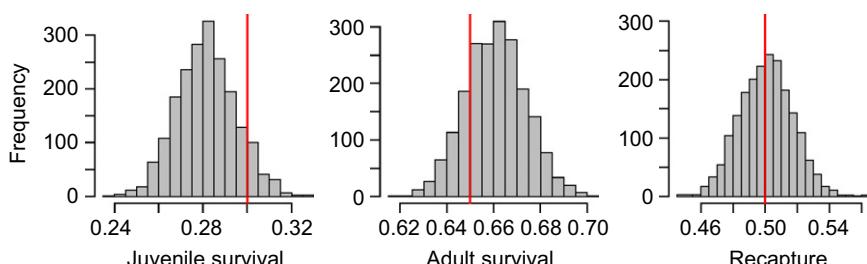
# Parameters monitored
parameters <- c("mean.phijuv", "mean.phiad", "mean.p")

# MCMC settings
ni <- 3000
nt <- 3
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
cjs.2 <- bugs(bugs.data, inits, parameters, "cjs-mnl-age.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

Convergence is achieved quickly; 3000 iterations with a burnin of 1000 are sufficient. Plotting the posterior distributions shows parameter estimates that resemble well the values used to simulate the data (Fig. 7.10).



**FIGURE 7.10** Posterior distributions of juvenile and adult survival and of recapture probability. Red lines indicate the values used to generate the data set.

```

par(mfrow = c(1, 3), las = 1)
hist(cjs.2$sims.list$mean.phijuv, nclass = 30, col = "gray", main = "",
     xlab = "Juvenile survival", ylab = "Frequency")
abline(v = phi.juv, col = "red", lwd = 2)
hist(cjs.2$sims.list$mean.phiad, nclass = 30, col = "gray", main = "",
     xlab = "Adult survival", ylab = "")
abline(v = phi.ad, col = "red", lwd = 2)
hist(cjs.2$sims.list$mean.p, nclass = 30, col = "gray", main = "",
     xlab = "Recapture", ylab = "")
abline(v = p[1], col = "red", lwd = 2)

```

We assumed that recapture probability was not dependent on age because all birds are >1 year old when they are first recaptured. Sometimes, however, it may be useful to fit age effects for recapture probability. Often, young individuals do not reproduce as successfully as adults. If individuals can only be captured when reproducing, this can result in a lower recapture probability of young individuals.

The model could also be extended to include more age classes. In principle, the number of m-arrays is equal to the number of age classes in the model. However, careful bookkeeping is required to fit these models. The age of each individual at each recapture has to be evaluated, and afterward the individual is “released” in the m-array in the corresponding age class. M-arrays are specified for each age class in the BUGS model code, and all of them have an age structure with the exception of the m-array for the oldest age class. Cell probabilities of the m-array of the second oldest age class have an age structure with two classes, that of the third-oldest age class an age structure with three classes, and so forth.

## 7.11 ANALYSIS OF A REAL DATA SET: SURVIVAL OF FEMALE LEISLER'S BATS

Leisler's bat (Fig. 7.11) is a medium-sized bat species that forms nursery colonies in cavities in woodlands and is widespread throughout Europe. Northern populations migrate to the Mediterranean in winter. Wigbert Schorcht and his colleagues studied a population of Leisler's bat in Thuringia (Germany) from 1989 to 2008. They placed bat boxes in a forest and regularly captured individuals in them. The capture–recapture data have been extensively analyzed using CJS models fitted in a frequentist framework (Schorcht et al., 2009). Here, we analyze a subset of these data consisting of 181 adult females that were born in the study area. Females are highly philopatric and thus our estimate of apparent survival is likely close to true survival. Some initial modeling suggested that adult survival was subject to strong temporal variation, whereas recapture probabilities were constant over time (Schorcht et al., 2009). Our interest here is to estimate mean annual survival as well as its temporal variance.



**FIGURE 7.11** Leisler's bat (*Nyctalus leisleri*) (Photograph D. Nill).

We will therefore fit a model denoted by  $(\phi_t, p_t)$ . We first performed a frequentist GOF test with program U-CARE (Choquet et al., 2001). The test assesses the fit of the time-dependent CJS model  $(\phi_t, p_t)$  and didn't show any indication of lack of fit ( $\chi^2_{49} = 41.38$ ,  $P = 0.77$ ). Yet, because we fit a model with random year effects, we want to perform also a posterior predictive check to evaluate the goodness-of-fit and estimate a Bayesian  $p$ -value for that model. The data are already summarized in the m-array format, thus we will use the multinomial likelihood to fit the model.

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,7,2,21,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,2,21,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,18), ncol = 19, nrow = 18,
byrow = TRUE)
```

The BUGS code poses no additional difficulties; we merely have to add the hierarchical extension to the multinomial model to account for random year effects. This extension assumes that the annual survival probabilities are random draws from a normal distribution whose mean is the logit of mean survival and a variance. This variance (`sigma2` in the code below) is the temporal variance of survival on the logit scale. In case we prefer to express the temporal variance on the probability scale, we also have a parameter called `sigma2.real`.

```
# Specify model in BUGS language
sink("cjs-mnl-ran.bug")
cat("
model {

# Priors and constraints
for (t in 1:(n.occasions-1)){
  logit(phi[t]) <- mu + epsilon[t]
  epsilon[t] ~ dnorm(0, tau)
  p[t] <- mean.p
}
mean.phi ~ dunif(0, 1)                      # Prior for mean survival
mu <- log(mean.phi / (1-mean.phi))          # Logit transformation
sigma ~ dunif(0, 5)                          # Prior for standard deviation
tau <- pow(sigma, -2)
sigma2 <- pow(sigma, 2)
# Temporal variance on real scale
sigma2.real <- sigma2 * pow(mean.phi, 2) * pow((1-mean.phi), 2)
mean.p ~ dunif(0, 1)                          # Prior for mean recapture

# Define the multinomial likelihood
for (t in 1:(n.occasions-1)){
  marr[t,1:n.occasions] ~ dmulti(pr[t,], r[t])
}

# Calculate the number of birds released each year
for (t in 1:(n.occasions-1)){
  r[t] <- sum(marr[t,])
}

# Define the cell probabilities of the m-array:
# Main diagonal
for (t in 1:(n.occasions-1)){
  q[t] <- 1-p[t]
  pr[t,t] <- phi[t]*p[t]
  # Above main diagonal
  for (j in (t+1):(n.occasions-1)){
    pr[t,j] <- prod(phi[t:j])*prod(q[t:(j-1)])*p[j]
  } #j
}
```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr[t,j]<-0
} #j
} #t
# Last column: probability of non-recapture
for (t in 1:(n.occasions-1)){
  pr[t,n.occasions] <- 1-sum(pr[t,1:(n.occasions-1)])
} # t

# Assess model fit using Freeman-Tukey statistic

# Compute fit statistics for observed data
for (t in 1:(n.occasions-1)){
  for (j in 1:n.occasions){
    expmarr[t,j] <- r[t]*pr[t,j]
    E.org[t,j] <- pow((pow(marr[t,j], 0.5)-pow(expmarr[t,j],
      0.5)), 2)
  }
}

# Generate replicate data and compute fit stats from them
for (t in 1:(n.occasions-1)){
  marr.new[t,1:n.occasions] ~ dmulti(pr[t,], r[t])
  for (j in 1:n.occasions){
    E.new[t,j] <- pow((pow(marr.new[t,j], 0.5)-pow(expmarr[t,j],
      0.5)), 2)
  }
}
fit <- sum(E.org[,])
fit.new <- sum(E.new[,])
}

",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(marr = m.leisleri, n.occasions = dim(m.leisleri) [2])

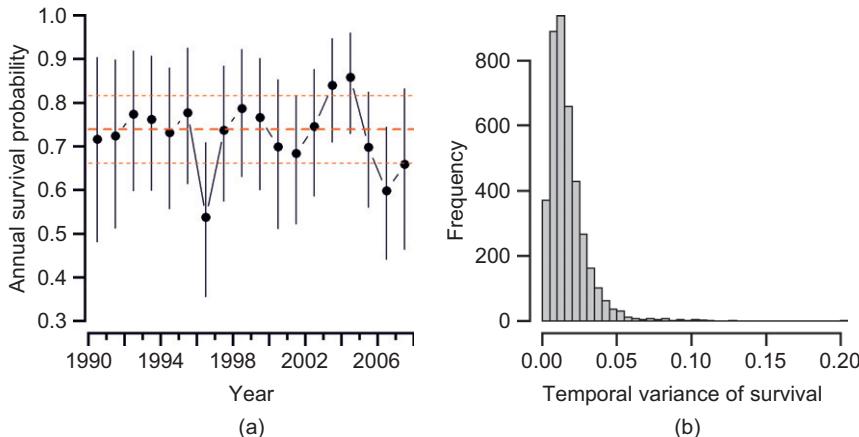
# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1), sigma = runif(1, 0,
  5), mean.p = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("phi", "mean.p", "mean.phi", "sigma2", "sigma2.real",
  "fit", "fit.new")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
leis.result <- bugs(bugs.data, inits, parameters, "cjs-mnl-ran.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```



**FIGURE 7.12** (a) Annual survival probability of adult female Leisler's bats (closed symbols, with 95% CRIs) and mean survival (red line; with 95% CRI dotted). (b) Posterior distribution of the temporal variance of adult survival.

The Markov chains converge quickly; with just 5000 iterations, we obtain satisfactory Rhat values (all  $< 1.01$ ). Mean annual survival is about 74%. Interestingly, and by chance, the recapture probability is numerically almost identical. Figure 7.12 plots the posterior distributions of the annual and mean survival probabilities as well as of the temporal variance. Annual survival probabilities were similar in most years, but in some, they were unusually low (1996–1997, 2006–2007) or unusually high (2003–2005). A next step in the demographic analysis of this population might be to find out which environmental factor is correlated with the temporal variation in survival, as shown in Section 7.4.3.

```
# Summarize posteriors
print(leis.result, digits = 3)

      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
phi[1] 0.716 0.106  0.481  0.653  0.726  0.789  0.904 1.001  4000
[ ... ]
phi[18] 0.658 0.093  0.464  0.600  0.661  0.723  0.832  1.004  590
mean.p 0.747 0.029  0.689  0.728  0.748  0.766  0.800  1.001  4000
mean.phi 0.739 0.038  0.661  0.715  0.739  0.763  0.815  1.003  2700
sigma2 0.467 0.340  0.062  0.235  0.386  0.610  1.341  1.012  390
sigma2.real 0.017 0.013  0.002  0.009  0.014  0.022  0.048  1.013  450
fit    21.047 2.279 17.260 19.410 20.850 22.400 26.279 1.003  830
fit.new 18.950 3.458 12.950 16.450 18.705 21.100 26.370 1.001  4000

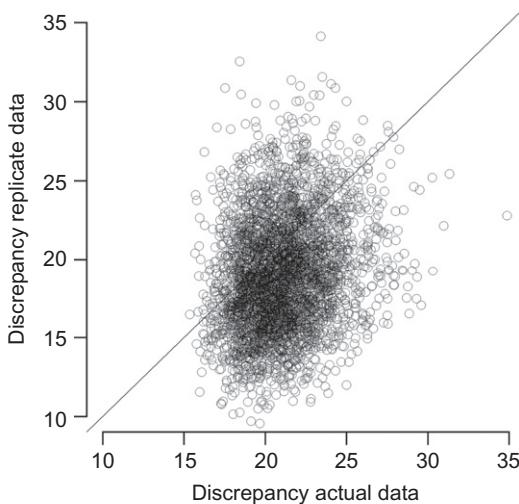
# Produce figure of female survival probabilities
par(mfrow = c(1, 2), las = 1, mar=c(4, 4, 2, 2), mgp = c(3, 1, 0))
lower <- upper <- numeric()
T <- dim(m.leisleri)[2]-1
```

```

for (t in 1:T) {
  lower[t] <- quantile(leis.result$sims.list$phi[,t], 0.025)
  upper[t] <- quantile(leis.result$sims.list$phi[,t], 0.975)
}
plot(y = leis.result$mean$phi, x = (1:T)+0.5, type = "b", pch = 16, ylim =
  c(0.3, 1), ylab = "Annual survival probability", xlab = "", axes = F)
axis(1, at = seq(1, (T+1), 2), labels = seq(1990, 2008, 2))
axis(1, at = 1:(T+1), labels = rep("", T+1), tcl = -0.25)
axis(2, las = 1)
mtext("Year", 1, line = 2.25)
segments((1:T)+0.5, lower, (1:T)+0.5, upper)
segments(1, leis.result$mean$mean.phi, T+1, leis.result$mean$mean.phi,
  lty = 2, col = "red", lwd = 2)
segments(1, quantile(leis.result$sims.list$mean.phi, 0.025), T+1,
  quantile(leis.result$sims.list$mean.phi, 0.025), lty = 2, col = "red")
segments(1, quantile(leis.result$sims.list$mean.phi, 0.975), T+1,
  quantile(leis.result$sims.list$mean.phi, 0.975), lty = 2, col = "red")
hist(leis.result$sims.list$sigma2.real, nclass = 45, col = "gray",
  main = "", las = 1, xlab = "")
mtext("Temporal variance of survival", 1, line = 2.25)

```

The GOF evaluation of the model shows a good fit (Fig. 7.13) with a Bayesian  $p$ -value of 0.27. The result is thus qualitatively the same as the GOF test performed in the frequentist framework (see above). Yet, the frequentist goodness-of-fit test evaluates the model with fixed year effects



**FIGURE 7.13** Scatter plot of replicate (simulated) versus actual (observed) discrepancy measures of model for female Leisler's bats. The Bayesian  $p$ -value is the proportion of points above the 1:1 equality line.

$(\phi_t, p_t)$ , whereas the Bayesian test evaluates the model actually used for the estimation, that is, a model with random year effects on survival and constant recapture probabilities  $(\phi_t, p.)$ .

```
# Evaluation of fit
plot(leis.result$sims.list$fit, leis.result$sims.list$fit.new,
      main = "", xlab = "Discrepancy actual data", ylab = "Discrepancy
      replicate data", las = 1, ylim = c(10, 35), xlim = c(10, 35), frame = FALSE)
abline(0, 1, col = "black")
```

## 7.12 SUMMARY AND OUTLOOK

This chapter presents models of the Cormack–Jolly–Seber (CJS) class for analysis of capture–recapture data in the Bayesian framework to estimate probabilities of survival and recapture. We introduced two different approaches, based on a state-space or a multinomial likelihood. The state-space likelihood has the advantage that it is very flexible and especially enables us to fit models with individual effects, including random effects. The downside is that the Markov chains of these models take much longer per iteration and mix less well, resulting sometimes in a big computational burden. With the multinomial likelihood, we cannot fit models with individual effects, but otherwise the same models are possible as under a state-space likelihood. Use of the multinomial likelihood results in quicker updates and better mixing of the chains. We therefore recommend using the multinomial likelihood unless individual effects need to be fitted.

This chapter contains very important material for the broad class of capture–recapture models because we have introduced several key concepts. We have shown how we can model survival (and recapture) along the “time” as well as along the “individual” axes using GLM formulations (see also Chapter 6 for the analogous concept to model detection probability). The corresponding models can have fixed or random effects, and there is great flexibility in combining them. In addition, we have introduced age-dependent models, which are a specific combination of effects along the time and the individual axes. We also have introduced goodness-of-fit testing using posterior predictive checks (Bayesian  $p$ -values). All these key concepts can be applied to the capture–recapture models in later chapters and indeed in an analogous way to all the models in the rest of the book.

Capture–recapture data could also be analyzed with the Jolly–Seber model (JS model; Williams et al., 2002), which is similar to the Cormack–Jolly–Seber model of this chapter. The main difference is that the CJS model conditions on first capture, whereas the JS model describes the complete capture-history. This means that the zeros before the first

capture are not modeled in the CJS model, but they are in the JS model. The latter allows the estimation of additional parameters such as recruitment and population size, at the expense of additional assumptions. We describe the JS model in Chapter 10. Further extensions to this class of model include the robust design model (Kendall et al., 1997; Schofield et al., 2009), reverse-time modeling to estimate population growth rate (Pradel, 1996), or the relative contribution of survival and recruitment to population growth (Nichols et al., 2000). These could be implemented in WinBUGS as well.

This chapter was the first to introduce models for estimation of survival and related demographic parameters. Much of the material (e.g., m-array, state-space likelihood, random and fixed effects in survival) also carries over to similar models in Chapters 8–10. In Chapter 11, we will combine the CJS model with other models into an integrated population model.

## 7.13 EXERCISES

1. For reasons of greater generality, we always specify CJS models with a likelihood that allows all parameters to potentially vary by individual and time. For a beginner, this may not be the simplest way to fit a CJS model. Consider the constant model in [Section 7.3](#) and adapt the BUGS model code so that we fit that model directly, without constraining the parameter matrices.
2. Simulate capture–recapture data of a species for males and females. The study is conducted for 15 years; the mean survival of males is 0.6 that of females is 0.5, and recapture is 0.4 for both. Assume that each year 30 individuals of each sex are newly marked. Fit the model  $\{\phi_{\text{sex}}, p\}$  to the data using the multinomial likelihood.
3. Simulate capture–recapture data of a species for males and females. The study is conducted for 10 years, and each year 30 young and 20 adults of each sex are newly marked. The mean survival of young males is 0.3 (0.2 for females) and mean survival of adults of both sexes is 0.7. Further assume that the recapture probability of males is time-dependent [0.5, 0.6, 0.4, 0.4, 0.7, 0.5, 0.8, 0.3, 0.8]. Recapture probability of females varies in parallel to that of the males, it is a bit higher than that of males (difference on the logit scale: 0.3). Analyze these data with the data-generating model.
4. For the model in [Section 7.3](#), do a simulation-based assessment of bias and precision. Generate a data set and then fit the model 500 times (perhaps for smaller sample size to save time) and each time save the estimates. On completion, print out the mean and the standard deviation of the estimates and also plot the distribution of these estimates. Is the estimator from the model biased? Where in the graph

can you see the standard error of the estimates? Are there other methods to check whether a model produces unbiased parameter estimates than simulation?

5. Take the data where survival of young and adult individuals is different ([Section 7.7](#)), but where only individuals of exact known age (marked as young) are included. Fit a model, in which survival after the second year changes linearly with increasing age.
6. Simulate data of a study that is running for 15 years, and each year 100 young individuals are marked. Survival in the first year is 0.4 on average with a temporal variability of 0.5 (on the logit scale), survival of older individuals is 0.8 without variability. Recapture probability is 0.6 for all individuals. Analyze these data with the data-generating model using the state-space and the multinomial likelihood.

# Estimation of Survival Using Mark-Recovery Data

## OUTLINE

8.1	Introduction	241
8.2	The Mark-Recovery Model as a State-Space Model	243
8.2.1	Simulation of Mark-Recovery Data	244
8.2.2	Analysis of a Model with Constant Parameters	246
8.3	The Mark-Recovery Model Fitted with the Multinomial Likelihood	248
8.3.1	Constant Parameters	248
8.3.2	Age-Dependent Parameters	252
8.4	Real-Data Example: Age-Dependent Survival in Swiss Red Kites	255
8.5	Summary and Outlook	261
8.6	Exercises	261

## 8.1 INTRODUCTION

Data on marked individuals that are reencountered alive formed the basis for the estimation of survival probabilities in the CJS models in Chapter 7. Here, we deal with another kind of data provided by marked individuals that serve the same goal: mark-recovery data. These data arise from animals that die and whose mark (typically a ring, but other marks are also possible) is recovered. As not all dead individuals will be found,

the probability that a marked, dead individual is found (the recovery probability) must be estimated in addition to the survival probability. Thus, the dead-recovery probability takes the place of the live detection probability in the CJS models.

The sampling protocol is as follows: a number of individuals are marked, preferably over several years or other defined periods, and information is collected about the time of death from the marked individuals, which typically comes from members of the public, who find and report dead individuals. For each reported individual, we know the place and year of death and thus can compute its longevity. For all marked individuals that are not found or reported, we do not know when they die. However, the number of marked individuals that are never recovered is known and can be expressed as a function of the same survival and recovery probabilities that act on the recovered individuals. Hence, individuals that are not recovered contribute information about survival as well.

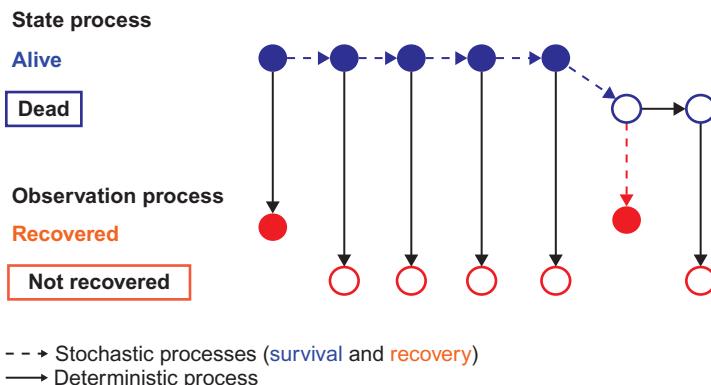
The unknown parameters are the survival probability ( $s$ ) and the recovery probability ( $r$ ). In theory, dead individuals can be found anywhere, not only in the study area. Hence, this survival probability is the true, rather than the apparent survival probability ( $\phi$ ) of the capture–recapture models in Chapter 7, where reencounters are usually restricted to fairly small areas. Apparent and true survival are linked by the fidelity probability ( $F$ , probability to remain within the study area), that is,  $\phi = sF$ . The recovery probability can be decomposed into a series of conditional binary events, such as the probability that an individual is found and the probability that the mark is reported. Sometimes, it is useful to consider them separately, resulting in a different parameterization of the mark-recovery model (Brownie et al., 1985).

Mark-recovery models are sometimes called dead-recovery or band-recovery models. They are primarily used for birds and in particular for hunted species because these have a much higher recovery probability than nongame species. The first treatise on these models was the handbook of Brownie et al. (1985), and a recent overview is provided in Williams et al. (2002). The Bayesian approach to these models is presented by Brooks et al. (2000a, 2000b, 2002), Barry et al. (2003), and Gimenez et al. (2007).

Here, we first show how the mark-recovery model can be fitted with a state-space formulation, the advantage of which is that we can apply exactly the same concepts (modeling along the time and individual axes; fixed and random effects) as introduced in Chapters 6 and 7; thus, we have great flexibility in modeling. However, we will not repeat these concepts here, but we just stress once more that all the key features introduced so far can be combined in a creative way. Second, we show the multinomial likelihood, which again enjoys great benefits in terms of computational efficiency.

## 8.2 THE MARK-RECOVERY MODEL AS A STATE-SPACE MODEL

Let us assume a newly marked individual at time  $t$ . It may survive until time  $t + 1$  with probability  $s_t$  ( $t = 1, \dots, T - 1$ ;  $T$  being the number of occasions). Conceptually, we can imagine the individual tossing a coin to determine whether it survives (with probability  $s_t$ ) or dies (with probability  $1 - s_t$ ). Given that the individual is still alive at time  $t + 1$ , it may again survive until  $t + 2$  with probability  $s_{t+1}$ . This is continued until either the individual is dead or the study ends. Clearly, once an individual is dead, its further fate is no longer a stochastic event, that is, it will remain dead with probability 1. What we have just described is the state process, that is, the possible states of an individual over time: dead or alive. As you may have noticed, this is exactly the same state process as in CJS models. The sole difference is a slightly different definition of the survival parameter: we use true survival in the ring-recovery model, whereas in the CJS model, it is the probability of surviving *and* remaining in the sampling area. The observation process for the mark-recovery model is different from the CJS model because only individuals that have just died can be observed (with probability  $r_t$ ). Once an individual has been dead for a while, it cannot be recovered anymore because it has typically decayed. Another assumption of the model is that the time of death of a recovered individual is known to the accuracy of the temporal interval of the occasions (typically 1 year). In Fig. 8.1, the state and the observation processes are shown graphically.



**FIGURE 8.1** Example of the state and observation process of a marked individual over time in the mark-recovery model. The sequence of true states in this individual is  $z = [1, 1, 1, 1, 0, 0]$ , and the observed capture-history is  $y = [1, 0, 0, 0, 0, 1, 0]$ . The occasion at which the individual is marked produces a “recovery”. This is just a convention to ensure that each capture-history starts with a 1.

The analysis of mark-recovery data with the state-space formulation is very similar to that of the CJS model. Again, we define the latent variable  $z_{i,t}$ , which takes value 1 if individual  $i$  is alive at time  $t$  and value 0 if it is dead. Thus,  $z_{i,t}$  denotes the true state of individual  $i$  at time  $t$ . We also define the vector  $f_i$  that contains the occasion at which individual  $i$  is marked (its first encounter). The state of individual  $i$  at first encounter  $z_{i,f_i}$  is 1 with probability 1, that is, the individual is alive with certainty. The states at subsequent occasions are Bernoulli trials: conditional on being alive at occasion  $t$ , individual  $i$  survives until occasion  $t+1$  with probability  $s_{i,t}$ . The following two equations define the state process:

$$\begin{aligned} z_{i,f_i} &= 1 \\ z_{i,t+1} \mid z_{i,t} &\sim \text{Bernoulli}(z_{i,t}s_{i,t}). \end{aligned}$$

The parameter of the Bernoulli trial is the product of the survival probability and the state variable  $z$ . The inclusion of  $z$  ensures that dead individuals (with  $z = 0$ ) remain dead.

Given that individual  $i$  is “recently dead” at occasion  $t$  (i.e., has died between  $t-1$  and  $t$ ), it may be recovered with probability  $r_{i,t}$  ( $t = 2, \dots, T$ ). The recovery, or observation, process is modeled as another Bernoulli trial with success probability  $r_{i,t}$ :

$$y_{i,t} \mid z_{i,t}, z_{i,t-1} \sim \text{Bernoulli}((z_{i,t-1} - z_{i,t})r_{i,t}).$$

The inclusion of the difference between the two successive values of latent variable  $z$  ensures that only individuals that are “recently dead” can be recovered. Matrix  $y$  contains the observed data, that is, the capture-histories. The state and the observation processes are both defined for  $t \geq f_i$ . The implementation of the model in the BUGS language now only requires the translation of these equations.

The mark-recovery model makes a number of assumptions that must be met to ensure unbiased parameter estimators. They are similar to those made in the CJS model. The survival and recovery probabilities are required to be identical for all individuals in the same group, cohort, or age class, and the fates of individuals must be independent. These assumptions can be tested with appropriate goodness-of-fit tests (Brownie et al., 1985). Furthermore, individuals must be reported without error (no misreading of the marks), and no mark loss is allowed.

### 8.2.1 Simulation of Mark-Recovery Data

We mimic a study of common terns (Fig. 8.2). Common terns live along rivers or coasts and breed in colonies, preferably on small islands. Adults are captured and marked, and some of them are found dead on migration



**FIGURE 8.2** Common terns (*Sterna hirundo*), Finland, 2004 (Photograph by J. Peltomäki).

or at the breeding grounds. We assume that over 14 years, we mark 50 adults every year. We assume annual adult survival of 0.8 and a recovery probability of 0.2. The following R code simulates a mark-recovery matrix. The initial capture and the recovery event are both denoted with 1. The mark-recovery data matrix has 15 columns because common terns can still be recovered after the last year of marking.

```

# Define parameter values
n.occasions <- 14                                # Number of release occasions
marked <- rep(50, n.occasions)                      # Annual number of marked individuals
s <- rep(0.8, n.occasions)
r <- rep(0.2, n.occasions)

# Define matrices with survival and recovery probabilities
S <- matrix(s, ncol = n.occasions, nrow = sum(marked))
R <- matrix(r, ncol = n.occasions, nrow = sum(marked))

# Define function to simulate mark-recovery data
simul.mr <- function(S, R, marked) {
  n.occasions <- dim(S) [2]
  MR <- matrix(NA, ncol = n.occasions+1, nrow = sum(marked))
  # Define a vector with the occasion of marking
  mark.occ <- rep(1:n.occasions, marked)
  # Fill the CH matrix
  for (i in 1:sum(marked)) {
    MR[i, mark.occ[i]] <- 1      # Write an 1 at the release occasion
    for (t in mark.occ[i]:n.occasions) {

```

```

# Bernoulli trial: has individual survived occasion?
sur <- rbinom(1, 1, S[i,t])
if (sur==1) next      # If still alive, move to next
                      occasion
# Bernoulli trial: has dead individual been recovered?
rp <- rbinom(1, 1, R[i,t])
if (rp==0){
  MR[i,t+1] <- 0
  break
}
if (rp==1){
  MR[i,t+1] <- 1
  break
}
} #t
} #i
# Replace the NA in the file by 0
MR[which(is.na(MR))] <- 0
return(MR)
}

# Execute function
MR <- simul.mr(S, R, marked)

```

### 8.2.2 Analysis of a Model with Constant Parameters

To fit the dead-recovery model, we also need a vector indicating the occasion of marking for each individual.

```

# Create vector with occasion of marking
get.first <- function(x) min(which(x!=0))
f <- apply(MR, 1, get.first)

```

We define the model and run the analysis.

```

# Specify model in BUGS language
sink("mr.ss.bug")
cat("
model {

# Priors and constraints
for (i in 1:nind) {
  for (t in f[i]:(n.occasions-1)) {
    s[i,t] <- mean.s
    r[i,t] <- mean.r
    } #t
  } #i
mean.s ~ dunif(0, 1)          # Prior for mean survival
mean.r ~ dunif(0, 1)          # Prior for mean recapture

# Likelihood
for (i in 1:nind) {

```

```

# Define latent state at first capture
z[i,f[i]] <- 1
for (t in (f[i]+1):n.occasions) {
  # State process
  z[i,t] ~ dbern(mu1[i,t])
  mu1[i,t] <- s[i,t-1] * z[i,t-1]
  # Observation process
  y[i,t] ~ dbern(mu2[i,t])
  mu2[i,t] <- r[i,t-1] * (z[i,t-1] - z[i,t])
} #t
} #i
},fill = TRUE)
sink()

```

For parameter estimation, we can either use the dead-recovery data matrix only or in addition provide information about the partly known latent state variable  $z$  (see Section 7.3.1). The latter results in faster computation and quicker convergence. In the dead-recovery model, the latent state variable is unknown for all individuals that are never recovered dead. However, for those that are recovered dead, the latent state is 1 for all occasions between marking and just prior to the recovery occasion and is 0 afterwards. The following function creates the known state variables from the dead-recovery matrix.

```

# Define function to create a matrix with information about known latent
state z
known.state.mr <- function(mr){
  state <- matrix(NA, nrow = dim(mr) [1], ncol = dim(mr) [2])
  rec <- which(rowSums(mr)==2)
  for (i in 1:length(rec)){
    n1 <- min(which(mr[rec[i],]==1))
    n2 <- max(which(mr[rec[i],]==1))
    state[rec[i],n1:n2] <- 1
    state[rec[i],n1] <- NA
    state[rec[i],n2:dim(mr) [2]] <- 0
  }
  return(state)
}

# Bundle data
bugs.data <- list(y = MR, f = f, nind = dim(MR) [1], n.occasions =
  dim(MR) [2], z = known.state.mr(MR))

# Define function to create a matrix of initial values for latent
state z
mr.init.z <- function(mr){
  ch <- matrix(NA, nrow = dim(mr) [1], ncol = dim(mr) [2])
  rec <- which(rowSums(mr)==1)
  for (i in 1:length(rec)){

```

```

n1 <- which(mr[rec[i],]==1)
ch[rec[i],n1:dim(mr)[2]] <- 0
ch[rec[i],n1] <- NA
}
return(ch)
}

# Initial values
inits <- function(){list(z = mr.init.z(MR), mean.s = runif(1, 0, 1),
mean.r = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.s", "mean.r")

# MCMC settings
ni <- 5000
nt <- 6
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 4 min)
mr.ss <- bugs(bugs.data, inits, parameters, "mr.ss.bug", n.chains =
nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
bugs.directory = bugs.dir)

```

The posterior means obtained with the state-space likelihood are nearly identical to those obtained with the multinomial likelihood (see Section 8.3). However, the state-space likelihood is computationally more demanding, and we need longer chains to achieve convergence.

```

print(mr.ss, digits = 3)
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat   n.eff
mean.s  0.754  0.027  0.699  0.737  0.755  0.773  0.802  1.008    400
mean.r  0.190  0.019  0.157  0.177  0.190  0.202  0.230  1.000   1500

```

As mentioned already, we can apply all the modeling encountered in the CJS within the framework of this basic model (i.e., the likelihood part). The changes required in the BUGS code are restricted to the model section entitled “Priors and constraints”.

## 8.3 THE MARK-RECOVERY MODEL FITTED WITH THE MULTINOMIAL LIKELIHOOD

### 8.3.1 Constant Parameters

The multinomial likelihood is the classical way to analyze mark-recovery data (Brownie et al., 1985; Williams et al., 2002). For this, the mark-recovery data are first summarized in the m-array (see Section 7.10). In this matrix, rows represent the release (marking) years, and the columns represent recovery years. The only difference between the mark-recovery and the

capture–recapture m-array is that in the former, all individuals that are released in a year (i.e., form a release cohort) remain in the same row of the matrix because they can be reencountered at most once; they are not released after recovery (difficult if they are dead ...). In the capture–recapture m-array, some individuals are released again in subsequent occasions and thus appear in more than one row of the matrix.

The following R code produces the necessary m-array for the mark-recovery data.

```
# Define function to create an m-array based for mark-recovery (MR) data
marray.dead <- function(MR) {
  nind <- dim(MR)[1]
  n.occasions <- dim(MR)[2]
  m.array <- matrix(data = 0, ncol = n.occasions+1, nrow =
    n.occasions)
  # Create vector with occasion of marking
  get.first <- function(x) min(which(x!=0))
  f <- apply(MR, 1, get.first)
  # Calculate the number of released individuals at each time period
  first <- as.numeric(table(f))
  for (t in 1:n.occasions) {
    m.array[t,1] <- first[t]
  }
  # Fill m-array with recovered individuals
  rec.ind <- which(apply(MR, 1, sum)==2)
  rec <- numeric()
  for (i in 1:length(rec.ind)) {
    d <- which(MR[rec.ind[i], (f[rec.ind[i]]+1):n.occasions]==1)
    rec[i] <- d+f[rec.ind[i]]
    m.array[f[rec.ind[i]],rec[i]] <- m.array[f[rec.ind[i]],
      rec[i]] + 1
  }
  # Calculate the number of individuals that are never recovered
  for (t in 1:n.occasions) {
    m.array[t,n.occasions+1] <- m.array[t,1]-sum(m.array[t,2:
      n.occasions])
  }
  out <- m.array[1:(n.occasions-1),2:(n.occasions+1)]
  return(out)
}
```

We produce the m-array for the simulated data by executing the function:

```
marr <- marray.dead(MR)
```

This m-array contains the observed response. Each row is modeled as a multinomial trial with index equal to the cohort size, that is, the number of individuals released in that year. The multinomial cell probabilities are functions of the parameters for survival and recovery ( $s$  and  $r$ ).

We first fit a model with constant parameters over time, but we have introduced an index of time (subscript) here in order to understand the model better. The cell probabilities are as follows:

Released at Occasion	Recovered at Occasion			Never Recovered
	2	3	4	
1	$(1 - s_1)r_1$	$s_1(1 - s_2)r_2$	$s_1s_2(1 - s_3)r_3$	$(1 - s_1)(1 - r_1) + s_1(1 - s_2)(1 - r_2) + s_1s_2(1 - s_3)(1 - r_3) + s_1s_2s_3 = 1 - \Sigma$ (Released at Occasion 1)
2	0	$(1 - s_2)r_2$	$s_2(1 - s_3)r_3$	$(1 - s_2)(1 - r_2) + s_2(1 - s_3)(1 - r_3) + s_2s_3 = 1 - \Sigma$ (Released at Occasion 2)
3	0	0	$(1 - s_3)r_3$	$(1 - s_3)(1 - r_3) + s_3 = 1 - \Sigma$ (Released at Occasion 3)

To fit this model in BUGS, we essentially have to define these cell probabilities. The probability for individuals never recovered may look complicated but is simply 1 minus the sum of the other probabilities in each row.

```
# Specify model in BUGS language
sink("mr-mnl.bug")
cat("
model {

# Priors and constraints
for (t in 1:n.occasions) {
  s[t] <- mean.s
  r[t] <- mean.r
}
mean.s ~ dunif(0, 1)           # Prior for mean survival
mean.r ~ dunif(0, 1)           # Prior for mean recovery

# Define the multinomial likelihood
for (t in 1:n.occasions) {
  marr[t,1:(n.occasions+1)] ~ dmulti(pr[t,,], rel[t])
}

# Calculate the number of birds released each year
for (t in 1:n.occasions) {
  rel[t] <- sum(marr[t,,])
}

# Define the cell probabilities of the m-array
# Main diagonal
for (t in 1:n.occasions) {
  pr[t,t] <- (1-s[t])*r[t]
  # Above main diagonal
  for (j in (t+1):n.occasions) {
    pr[t,j] <- prod(s[t:(j-1)])*(1-s[j])*r[j]
  } #j
}
```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr[t,j] <- 0
} #j
} #t
# Last column: probability of non-recovery
for (t in 1:n.occasions){
  pr[t,n.occasions+1] <- 1-sum(pr[t,1:n.occasions])
} #t
}
",fill = TRUE)
sink()

```

Now, we run the mark-recovery model using the m-array data.

```

# Bundle data
bugs.data <- list(marr = marr, n.occasions = dim(marr) [2]-1)

# Initial values
inits<- function() {list(mean.s = runif(1, 0, 1), mean.r = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.s", "mean.r")

# MCMC settings
ni <- 5000
nt <- 6
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
mr <- bugs(bugs.data, inits, parameters, "mr-mnl.bug", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir)

# Summarize posteriors
print(mr, digits = 3)
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat   n.eff
mean.s  0.758  0.033  0.691  0.736  0.757  0.782  0.823  1.003    910
mean.r  0.192  0.020  0.156  0.179  0.191  0.204  0.236  1.001   1500

```

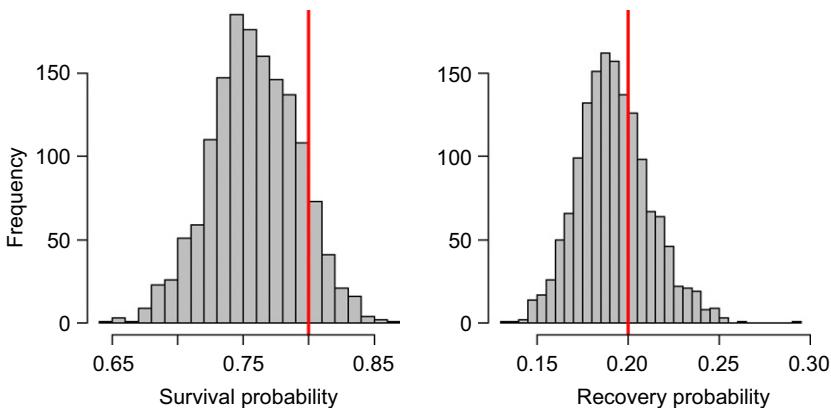
Convergence of the chains is not difficult to achieve. The posterior distributions of both parameters include the input values of 0.8 and 0.2 (Fig. 8.3).

```

par(mfrow = c(1, 2), las = 1)
hist(mr$sims.list$mean.s, nclass = 25, col = "gray", main = "", ylab =
  "Frequency", xlab = "Survival probability")
abline(v = 0.8, col = "red", lwd = 2)
hist(mr$sims.list$mean.r, nclass = 25, col = "gray", main = "", ylab =
  "", xlab = "Recovery probability")
abline(v = 0.2, col = "red", lwd = 2)

```

It is straightforward to build more complicated models, for example, with different groups, or fixed or random temporal variation. The BUGS



**FIGURE 8.3** Posterior distributions of survival and recovery probabilities in the analysis of the simulated data (red line: values used for simulating the data).

code just has to be adapted in exactly the same way as explained for the capture–recapture models (Chapter 7). Basically, the only changes are made in the model section entitled “Priors and constraints”.

### 8.3.2 Age-Dependent Parameters

Survival typically changes with age; hence, age-dependent models are often important. Sometimes, even recovery probability may change with age. Age-dependent recovery probabilities can occur if the main mortality cause changes with age, and different mortality causes are associated with differential recovery probabilities. For example, individuals dying from human-related causes are more likely found by humans than individuals dying from natural causes. As an example, the recovery probability of white storks (*Ciconia ciconia*) dying from power line collisions is significantly higher than for other causes of mortality (Schaub and Pradel, 2004).

We stay with our common tern example, and assume that young (nestlings) and adults are marked. We assume survival to be 0.3 for juveniles and 0.8 for adults. The recovery probabilities for juveniles and adults are 0.25 and 0.15, respectively. We simulate two data sets: one for individuals marked as young and another for individuals marked as adults. We summarize both data sets independently as an m-array.

```

n.occasions <- 15 # Number of occasions
marked.j <- rep(200, n.occasions) # Annual number of newly marked
# young
marked.a <- rep(20, n.occasions) # Annual number of newly marked
# adults
sjuv <- 0.3 # Juvenile survival probability
sad <- 0.8 # Adult survival probability

```

```

rjuv <- 0.25                      # Juvenile recovery probability
rad <- 0.15                        # Adult recovery probability
sj <- c(sjuv, rep(sad, n.occasions-1))
rj <- c(rjuv, rep(rad, n.occasions-1))

# Define matrices with survival and recovery probabilities
SJ <- matrix(0, ncol = n.occasions, nrow = sum(marked.j))
for (i in 1:length(marked.j)){
  SJ[(sum(marked.j[1:i])-marked.j[i]+1) : sum(marked.j[1:i]),
      i:n.occasions] <- matrix(rep(sj[1:(n.occasions-i+1)],
                                    marked.j[i]), ncol = n.occasions-i+1, byrow = TRUE)
}
SA <- matrix(sad, ncol = n.occasions, nrow = sum(marked.a))
RJ <- matrix(0, ncol = n.occasions, nrow = sum(marked.j))
for (i in 1:length(marked.j)){
  RJ[(sum(marked.j[1:i])-marked.j[i]+1) : sum(marked.j[1:i]),
      i:n.occasions] <- matrix(rep(rj[1:(n.occasions-i+1)],
                                    marked.j[i]), ncol = n.occasions-i+1, byrow = TRUE)
}
RA <- matrix(rad, ncol = n.occasions, nrow = sum(marked.a))

# Execute simulation function
MRj <- simul.mr(SJ, RJ, marked.j)
MRA <- simul.mr(SA, RA, marked.a)

# Summarize data in m-arrays
marr.j <- marray.dead(MRj)
marr.a <- marray.dead(MRA)

```

The cell probabilities in the modeling of the m-array for adults are exactly the same as before except for  $s_i$  and  $r_i$ , which we now denote  $sa_i$  and  $ra_i$ , respectively. In contrast, the cell probabilities for the model of individuals marked as nestlings differ: for them, we have to include an age structure for the survival and recovery probabilities. These cell probabilities are as follows:

Released at Occasion	Recovered at Occasion			Never Recovered
	2	3	4	
1	$(1 - sj_1)rj_1$	$sj_1(1 - sa_2)ra_2$	$sj_1sa_2(1 - sa_3)ra_3$	$1 - \Sigma (\text{Released at Occasion } 1)$
2	0	$(1 - sj_2)rj_2$	$sj_2(1 - sa_3)ra_3$	$1 - \Sigma (\text{Released at Occasion } 2)$
3	0	0	$(1 - sj_3)rj_3$	$1 - \Sigma (\text{Released at Occasion } 3)$

In the model, we define the probabilities of both m-arrays separately, with some parameters ( $sa$  and  $ra$ ) shared.

```

# Specify model in BUGS language
sink("mr-mnl-age.bug")
cat("
model {

```

```

# Priors and constraints
for (t in 1:n.occasions) {
  sj[t] <- mean.sj
  sa[t] <- mean.sa
  rj[t] <- mean.rj
  ra[t] <- mean.ra
}
mean.sj ~ dunif(0, 1)          # Prior for mean juv.survival
mean.sa ~ dunif(0, 1)          # Prior for mean ad.survival
mean.rj ~ dunif(0, 1)          # Prior for mean juv.recovery
mean.ra ~ dunif(0, 1)          # Prior for mean ad.recovery

# Define the multinomial likelihoods
for (t in 1:n.occasions) {
  marr.j[t,1:(n.occasions+1)] ~ dmulti(pr.j[t,], rel.j[t])
  marr.a[t,1:(n.occasions+1)] ~ dmulti(pr.a[t,], rel.a[t])
}

# Calculate the number of birds released each year
for (t in 1:n.occasions) {
  rel.j[t] <- sum(marr.j[t,])
  rel.a[t] <- sum(marr.a[t,])
}

# Define the cell probabilities of the juvenile m-array
# Main diagonal
for (t in 1:n.occasions) {
  pr.j[t,t] <- (1-sj[t])*rj[t]
  # Further above main diagonal
  for (j in (t+2):n.occasions) {
    pr.j[t,j] <- sj[t]*prod(sa[(t+1):(j-1)])*(1-sa[j])*ra[j]
    } #j
  # Below main diagonal
  for (j in 1:(t-1)) {
    pr.j[t,j] <- 0
    } #j
} #t
for (t in 1:(n.occasions-1)) {
  # One above main diagonal
  pr.j[t,t+1] <- sj[t]*(1-sa[t+1])*ra[t+1]
  } #t
# Last column: probability of non-recovery
for (t in 1:n.occasions) {
  pr.j[t,n.occasions+1] <- 1-sum(pr.j[t,1:n.occasions])
  } #t

# Define the cell probabilities of the adult m-array
# Main diagonal
for (t in 1:n.occasions) {
  pr.a[t,t] <- (1-sa[t])*ra[t]
  # Above main diagonal
  for (j in (t+1):n.occasions) {
    pr.a[t,j] <- prod(sa[t:(j-1)])*(1-sa[j])*ra[j]
    } #j
}

```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr.a[t,j] <- 0
} #j
} #t
# Last column: probability of non-recovery
for (t in 1:n.occasions){
  pr.a[t,n.occasions+1] <- 1-sum(pr.a[t,1:n.occasions])
} #t
}
",fill = TRUE)
sink()

```

We prepare the remainder of the analysis and run the mark-recovery model.

```

# Bundle data
bugs.data <- list(marr.j = marr.j, marr.a = marr.a, n.occasions =
  dim(marr.j) [2]-1)

# Initial values
inits <- function(){list(mean.sj = runif(1, 0, 1), mean.sa =
  runif(1, 0, 1), mean.rj = runif(1, 0, 1), mean.ra = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("mean.sj", "mean.rj", "mean.sa", "mean.ra")

# MCMC settings
ni <- 5000
nt <- 6
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
mr.age <- bugs(bugs.data, inits, parameters, "mr-mnl-age.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug =
  TRUE, bugs.directory = bugs.dir)

```

As before, convergence is achieved fairly quickly.

```

print(mr.age, digits = 3)
      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
mean.sj  0.289  0.051  0.205  0.252  0.283  0.319  0.402  1.000  1500
mean.rj  0.254  0.022  0.219  0.239  0.250  0.265  0.304  1.001  1500
mean.sa  0.797  0.025  0.748  0.779  0.797  0.814  0.848  1.000  1500
mean.ra  0.192  0.029  0.140  0.171  0.190  0.210  0.251  1.000  1500

```

## **8.4 REAL-DATA EXAMPLE: AGE-DEPENDENT SURVIVAL IN SWISS RED KITES**

The red kite (Fig. 8.4) is a large bird of prey that is declining in many parts of Europe but widespread in the Swiss lowlands. During the past 50 years, 1480 nestlings and 152 adults (>2 years old) have been marked,



**FIGURE 8.4** Red kite (*Milvus milvus*), Switzerland (Photograph by P. Keusch).

of which 107 individuals were recovered dead. Our interest was to estimate age-specific survival probabilities; we were not interested in any changes over time. Therefore, we considered all individuals marked at the same age as a single release cohort. Here, we have two age classes at marking: juveniles and adults. Since we drop the time index, the resulting m-array of both age classes consists only of a single line, with “columns” referring to the age at which individuals are recovered. Strictly speaking, only in the m-array for individuals marked as juveniles do the columns correspond to true age because in these individuals true age is known. In the m-array of the adults, columns refer to the age since marking (in years) because the age of these individuals at marking is unknown. For example, in the data given below, we see that 1388 of all individuals marked as juveniles are never found, whereas five individuals are found dead in their third year of life. Here are the data for the individuals marked as juveniles and those marked as adults.

```
marray.juv <- c(42, 18, 5, 7, 4, 3, 2, 1, 2, 2, 1, 0, 1, 3, 0, 0, 1, 1388)
marray.ad <- c(3, 1, 1, 3, 0, 2, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 137)
```

In long-lived birds, survival usually varies with age. Here, we fit a survival model with three age classes. The first age class (juveniles) covers one year from fledging, the second age class (subadults) also covers one

year from one to two years of age, and the last age class (adults) contains all ages 2 years and greater. For recovery probability, we only assume two age classes and distinguish the first year after fledging from all ages thereafter. Juvenile survival in mark-recovery models is sometimes difficult to estimate (see discussion below), and therefore, we will also try to utilize a priori information on juvenile survival. Few estimates of juvenile survival in red kites are available, and most of them were obtained using dated methods, that is, ignoring imperfect detection. On average, juvenile survival is likely to be about 0.6 but with a large uncertainty (Aebischer, 2009). We translated this knowledge into a beta prior distribution with parameters 4.2 and 2.8. This ensured a mean of 0.6 and a relatively wide spread of the prior (standard deviation  $\approx 0.173$ ). To assess prior sensitivity, we also fitted a model with a flat uniform prior ( $U(0,1)$ ). The BUGS code above needs some adaptations because we now have three age classes, but only two release cohorts: juveniles and adults.

```
# Specify model in BUGS language
sink("mr-mnl-age3.bug")
cat("
model {

# Priors and constraints
sjuv ~ dbeta(4.2, 2.8)          # Informative prior for juv. survival:
                                 # Analysis A
#sjuv ~ dunif(0, 1)             # Non-informative for juv. survival prior:
                                 # Analysis B
ssub ~ dunif(0, 1)               # Prior for subad. survival
sad ~ dunif(0, 1)                 # Prior for ad. survival
rjuv ~ dunif(0, 1)                # Prior for juv. recovery
rad ~ dunif(0, 1)                 # Prior for ad. recovery

# Define the multinomial likelihoods
marr.j[1:(n.age+1)] ~ dmulti(pr.j[], rel.j)
marr.a[1:(n.age+1)] ~ dmulti(pr.a[], rel.a)

# Calculate the number of birds released each year
rel.j <- sum(marr.j[])
rel.a <- sum(marr.a[])

# Define the cell probabilities of the juvenile m-array
# First element
pr.j[1] <- (1-sjuv)*rjuv
# Second element
pr.j[2] <- sjuv*(1-ssub)*rad
# Third and further elements
for (t in 3:n.age){
  pr.j[t] <- sjuv*ssub*pow(sad, (t-3))*(1-sad)*rad
}
# Probability of non-recovery
pr.j[n.age+1] <- 1 - sum(pr.j[1:n.age])
```

```

# Define the cell probabilities of the adult m-array
# All elements
for (t in 1:n.age) {
  pr.a[t] <- pow(sad, (t-1))*(1-sad)*rad
}
# Probability of non-recovery
pr.a[n.age+1] <- 1-sum(pr.a[1:n.age])
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(marr.j = marray.juv, marr.a = marray.ad, n.age =
  length(marray.juv)-1)

# Initial values
inits <- function(){list(sjuv = runif(1, 0, 1), ssub = runif(1, 0, 1),
  sad = runif(1, 0, 1), rjuv = runif(1, 0, 1), rad = runif(1, 0, 1))}

# Parameters monitored
parameters <- c("sjuv", "ssub", "sad", "rjuv", "rad")

# MCMC settings
ni <- 30000
nt <- 10
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT <1 min)
rk.ageA <- bugs(bugs.data, inits, parameters, "mr-mnl-age3.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug =
  TRUE, bugs.directory = bugs.dir)

```

Convergence is reached quickly. Here are the posterior summaries for the estimated parameters.

```

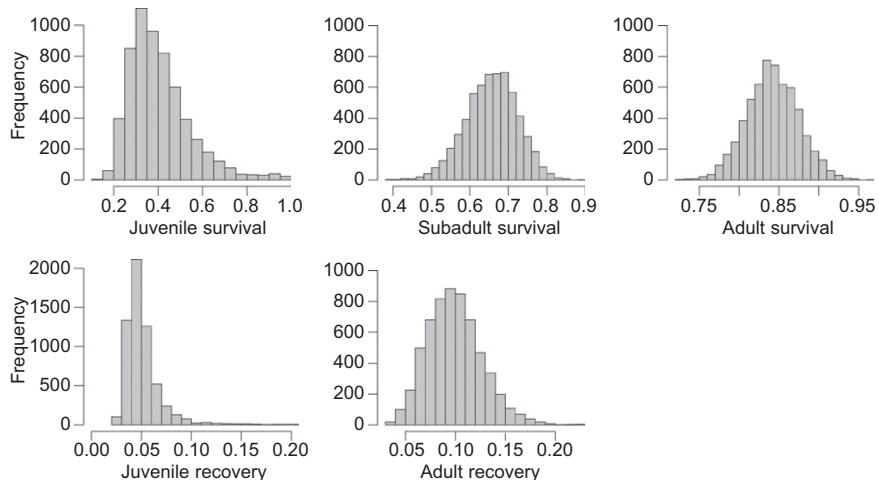
print(rk.ageA, digits = 3)
      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
sjuv  0.450  0.128  0.248  0.358  0.431  0.525  0.751  1.003 1100
ssub  0.655  0.067  0.521  0.611  0.658  0.702  0.780  1.001 2800
sad   0.841  0.032  0.778  0.820  0.842  0.863  0.904  1.001 4900
rjuv  0.057  0.027  0.033  0.043  0.051  0.062  0.119  1.002 1700
rad   0.090  0.024  0.051  0.073  0.088  0.104  0.143  1.002 1700

```

As shown below, parameter estimates under the model with noninformative priors are similar (for this, we repeat the analysis with one prior statement switched) to those under the model with informative priors. Two exceptions are the posterior mean of juvenile survival (lower) and the standard deviation of the juvenile recovery probability (higher). Thus, the prior information introduced for juvenile survival affected also juvenile recovery probability (and even adult recovery probability). This is not very surprising.

```
print(rk.ageB, digits = 3)
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
sjuv  0.406  0.139  0.216  0.309  0.378  0.473  0.773  1.003  920
ssub  0.656  0.067  0.518  0.612  0.660  0.703  0.779  1.001  6000
sad   0.842  0.032  0.779  0.820  0.842  0.864  0.906  1.001  6000
rjuv  0.057  0.060  0.031  0.040  0.047  0.057  0.129  1.001  3500
rad   0.099  0.027  0.052  0.080  0.097  0.116  0.159  1.003  1000
```

Inspection of the posterior distributions of the five parameters under the model with noninformative priors shows that subadult and adult survival and adult recovery probability are estimated precisely (Fig. 8.5). By contrast, the posterior distribution of juvenile survival is quite wide and has some of its mass extending to 1, whereas that of juvenile recovery is skewed. This does not mean that we cannot trust the estimates; it simply indicates that there is considerable uncertainty about the values of these parameters. This behavior is due to intrinsic identifiability problems. When survival and recovery probabilities are age dependent and only data of individuals marked as juveniles are available, only adult survival is identifiable (Anderson et al., 1985). Here, we have a different number of age classes for survival and recovery, and we included also data on individuals marked as adults; thus, the parameters in the model are intrinsically identifiable. Yet, from the posterior distributions of juvenile survival in particular, we see that the available information is a bit scarce.



**FIGURE 8.5** Posterior distributions of survival and recovery probabilities of Swiss red kites under model B with noninformative priors.

```

par(mfrow = c(2, 3), las = 1)
hist(rk.ageB$sims.list$sjuv, breaks = 20, col = "gray", main = "",
      xlab = "Juvenile survival")
hist(rk.ageB$sims.list$ssub, breaks = 20, col = "gray", main = "",
      xlab = "Subadult survival")
hist(rk.ageB$sims.list$sad, breaks = 20, col = "gray", main = "",
      xlab = "Adult survival")
hist(rk.ageB$sims.list$rjuv, breaks = 20, col = "gray", main = "",
      xlab = "Juvenile recovery", xlim = c(0, 0.2))
hist(rk.ageB$sims.list$rad, breaks = 20, col = "gray", main = "",
      xlab = "Adult recovery")

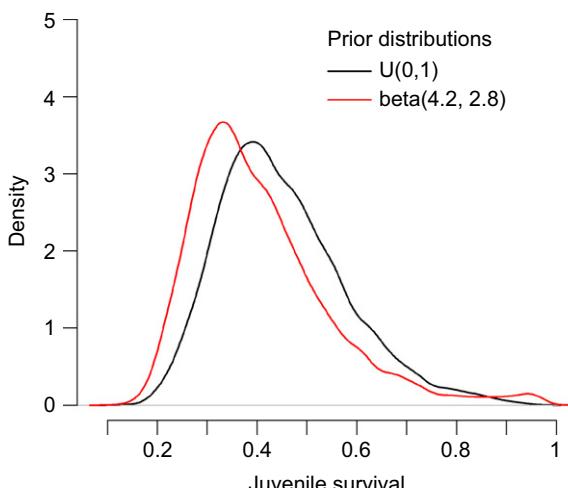
```

Comparison of the posterior distributions of juvenile survival under the two priors shows that both are similar (Fig. 8.6). Thus, existing information from the literature did not have a very strong impact, and the uncertainty about juvenile survival was not much reduced. We would just like to note that the specification of informative priors can be a challenge, and there may be better ways than the ad hoc approach in our example (e.g., McCarthy and Masters, 2005; Servanthy et al., 2010).

```

plot(density(rk.ageA$sims.list$sjuv), ylim = c(0, 5), lwd = 2,
      main = "", xlab = "Juvenile survival", las = 1)
points(density(rk.ageB$sims.list$sjuv), col = "red", type = "l",
      lwd = 2)
text(x = 0.5, y = 4.8, "Prior distributions", pos = 4, font = 3)
legend(x = 0.6, y = 4.7, legend = c("U(0,1)", "beta(4.2, 2.8)"),
      lwd = c(2, 2), col = c("black", "red"), bty = "n")

```



**FIGURE 8.6** Posterior distributions of juvenile red kite survival with an informative ( $\text{beta}(4.2, 2.8)$ , red) and a noninformative ( $\text{U}(0, 1)$ , black) prior.

Finally, we reiterate the ease with which derived quantities are estimated in a Bayesian framework with posterior sampling: just compute the derived quantity for every MCMC iteration and summarize that new MCMC sample. For instance, here are the 95% CRI for the differences between juvenile and subadult and between subadult and adult survival:

```
quantile(rk.ageA$sims.list$ssub-rk.ageA$sims.list$sjuv, prob =
  c(0.025, 0.975))
  2.5%      97.5%
-0.1106200   0.4867075

quantile(rk.ageA$sims.list$sd-rk.ageA$sims.list$ssub, prob =
  c(0.025, 0.975))
  2.5%      97.5%
  0.0595925   0.3242150
```

## 8.5 SUMMARY AND OUTLOOK

We have introduced an important class of models for estimation of survival and recovery probabilities from mark-recovery data, typically for birds. We have shown two different parameterizations of the model. The model based on the multinomial likelihood is computationally cheaper and can be extended to include groups, temporal covariates, as well as age classes. The model based on the state-space likelihood is required if we want to fit individual covariates, but this comes at the expense of much longer computational time. The combination of age-effects and individual covariates is best done with multistate models (Chapter 9).

Mark-recovery models have a long history, and many variants exist. They include models for the estimation of seasonal survival probabilities (Tavecchia et al., 2002), kill rates (Nichols et al., 1991), the proportion of animals dying from different mortality causes (Schaub and Pradel, 2004), and spatial variation of recovery probabilities (Royle and Dubovsky, 2001), to name just a few. Williams et al. (2002) provide a recent review of mark-recovery models and include plenty of additional information, such as different parameterizations, study design, or goodness-of-fit testing.

Mark-recovery data can also be analyzed with multistate models (Lebreton et al., 1999; Gauthier and Lebreton, 2008; see also Section 9.5). Moreover, it is possible to jointly analyze mark recovery and capture-recapture data, although they provide different measures of survival. Multistate capture–recapture models are the best tool to conduct such a joint analysis (see Section 9.5).

## 8.6 EXERCISES

1. Simulate mark-recovery data of two groups: both groups have a survival probability of 0.5, the first group has a recovery probability of 0.1, and the second group has a recovery probability of 0.2. The study is

- conducted for 10 years, and each year, 50 individuals are marked in each group. Fit the model  $(s, r_g)$  using (1) the multinomial and (2) the state-space likelihoods.
2. It is quite typical for population studies that only nestlings are marked but no adult individuals. This is because the capture of adults is often much more time consuming than the marking of nestlings, which can be easily marked in the nest. Simulate data from a study on a common tern population in which only nestlings are marked. The study duration is 15 years; in each year, 200 nestlings are marked, and the parameters are  $sj = 0.3$ ,  $sa = 0.8$ ,  $rj = 0.25$ , and  $ra = 0.15$ . Analyze these data with (1) the data-generating model and (2) using a model in which the recovery probability is the same in both age classes. Comment on the parameter estimates that you obtain from both models.
3. Simulate mark-recovery data with the following characteristics: one group, during each of the 20 study years 500 individuals are released, the survival probability declines linearly from 0.8 in the first year to 0.6 in the last study year, whereas the recovery probability is constant at 0.05. Analyze these data with the multinomial model.
4. Because of differential behavior, the recovery probability may show strong individual variation. Simulate mark-recovery data for a population with mean survival of 0.7 and a mean recovery probability of 0.2. The variance of the recovery probability among individuals is 0.7 (on the logit scale). Assume that the study lasts 10 years and that each year 100 individuals are released. Analyze the data with (1) the data-generating model and (2) with a model that assume a common recovery probability for all individuals. What is the impact on the estimate of the survival probability?

## 9

# Estimation of Survival and Movement from Capture–Recapture Data Using Multistate Models

## OUTLINE

9.1	Introduction	264
9.2	Estimation of Movement between Two Sites	268
9.2.1	Model Description	268
9.2.2	Generation of Simulated Data	270
9.2.3	Analysis of the Model	274
9.3	Accounting for Temporary Emigration	281
9.3.1	Model Description	281
9.3.2	Generation of Simulated Data	282
9.3.3	Analysis of the Model	284
9.4	Estimation of Age-Specific Probability of First Breeding	288
9.4.1	Model Description	288
9.4.2	Generation of Simulated Data	289
9.4.3	Analysis of the Model	290
9.5	Joint Analysis of Capture–Recapture and Mark-Recovery Data	295
9.5.1	Model Description	295
9.5.2	Generation of Simulated Data	296
9.5.3	Analysis of the Model	297

<b>9.6 Estimation of Movement among Three Sites</b>	<b>300</b>
9.6.1 Model Description	300
9.6.2 Generation of Simulated Data	302
9.6.3 Analysis of the Model	304
<b>9.7 Real-Data Example: The Showy Lady's Slipper</b>	<b>307</b>
<b>9.8 Summary and Outlook</b>	<b>311</b>
<b>9.9 Exercises</b>	<b>312</b>

## 9.1 INTRODUCTION

When an individual is encountered, it is often possible to assign it to a certain *state*. A state may be described as a categorical individual covariate that can change over time. A state may be a geographical location, a class of reproductive success, or a disease status, to name just a few. Stratification of individuals according to their state then allows estimation of state-dependent survival and recapture probabilities, as well as state-transition probabilities. In this chapter, we extend the seminal CJS model from Chapter 7 to more than a single state. Perhaps not surprisingly, the resulting model is called a multistate (or historically also multistrata) model (Lebreton et al., 2009).

Depending on how the states are defined, a vast array of research questions can be addressed using this broad class of capture–recapture models. For example, if states represent geographic locations, we can estimate movement probabilities among locations, that is, dispersal. We may test whether movement is symmetrical or is related to habitat quality of a location. If states represent “not infected by a disease” and “infected by a disease”, we may study whether survival is related to disease status or identify conditions that are positively correlated with the infection probability, that is, the transition probability from “not infected” to “infected”. State definitions may also be less obvious. States could be age classes, which allows estimation of age-dependent survival probabilities, they could be “dead” and “alive” enabling modeling of mark-recovery data or the joint analysis of capture–recapture and mark-recovery data, they could be “present” and “absent” allowing estimation of temporary emigration, or they could be “captured” and “not captured”, which enables fitting of models with immediate trap response. In fact, multistate capture–recapture models are extremely useful and should be used for many interesting ecological

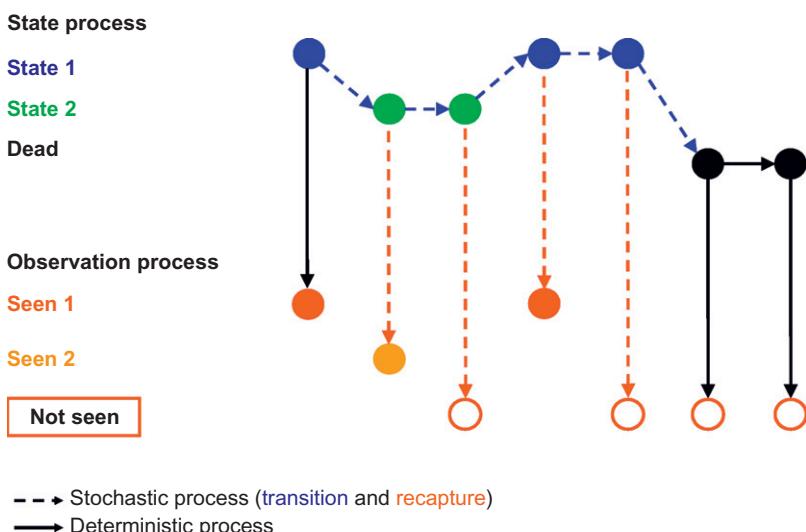
questions. Multistate models represent a unification of many different kinds of capture–recapture model (Lebreton et al., 1999, 2009).

There are many applications of multistate models in the frequentist framework. As an example, see the seminal papers by Arnason (1972, 1973), Hestbeck et al. (1991), Schwarz et al. (1993), Lebreton and Pradel (2002) and the review by Lebreton et al. (2009). Much fewer authors have adopted the Bayesian framework so far (e.g., Dupuis, 1995; King and Brooks, 2002; Gimenez et al., 2003; King and Brooks, 2004; Clark et al., 2005; Dupuis and Schwarz, 2007; Calvert et al., 2009; Schofield et al., 2009; King et al., 2010; Schaub et al., 2010).

As the single-state capture–recapture and mark-recovery models in Chapters 7 and 8, multistate capture–recapture models can be analyzed either in a state-space modeling framework or using a multinomial likelihood. Frequentist analyses typically use the multinomial likelihood applied to the multistate m-array (Williams et al., 2002). This approach requires definition of all cell probabilities in the multistate m-array, which is straightforward when matrix multiplication is available in a software. Unfortunately, matrix multiplication is not available in WinBUGS (though it is in JAGS); hence, we will only use the state-space formulation here. It has the advantage that exactly the same concepts (modeling along the time and individual axis, with fixed and random effects) as introduced in Chapters 6 and 7 can be applied providing great flexibility in modeling. We will not repeat these concepts in this chapter anymore, but refer to Chapters 6 and 7 and to the exercises. The state equation of the state-space formulation describes the true development of the states, that is, it describes the state of an individual at time  $t + 1$ , given its state at time  $t$ . The observation equation maps the true state at time  $t$  on to the observed state. Since there are more than two possible true and observed states, the likelihood is not based on the Bernoulli distribution, as is the case with single-state capture–recapture models, but on the categorical distribution. The categorical distribution is the extension of a Bernoulli distribution to more than two categories. It is a special case of the multinomial distribution with trial size equal to 1.

For a description of the state process in a system with two geographic locations, let us assume that a marked individual is at location 1 (referred to as state 1) at time  $t$ . At the next sampling occasion,  $t + 1$ , this individual may still be alive, and at location 1 with probability  $\Phi_{11,t}$ , it may still be alive but has moved to location 2 (state 2) with probability  $\Phi_{12,t}$ , or it may be dead with probability  $1 - \Phi_{11,t} - \Phi_{12,t}$ . If the individual is still alive at time  $t + 1$ , it may again survive and move among locations. If the individual is at location 2 at time  $t + 1$ , then the corresponding probability to survive and move to location 1 is  $\Phi_{21,t+1}$ , the probability to survive and remain at location 2 is  $\Phi_{22,t+1}$ , and the probability to die is the complement

of the two, that is,  $1 - \Phi_{21,t+1} - \Phi_{22,t+1}$ . This is continued until the individual is either dead or a study ends. Once an individual is dead, its fate is no longer stochastic; the individual will remain dead with probability 1 until the end of the study. These probabilities define the state process, and we would like to know the parameters governing this process. Yet, inevitably, we cannot observe the state process without errors. Instead, we augment our system description with a component that describes the observation process. A marked individual that is alive at time  $t$  in state 1 may be encountered in state 1 with probability  $p_{1,t}$ , whereas an individual that is alive at time  $t$  and in state 2 may be encountered in state 2 with probability  $p_{2,t}$ . We can imagine that for each individual and occasion, a coin is tossed determining whether the individual is encountered (with probability  $p_{1,t}$  or  $p_{2,t}$ ) or not (with probability  $1 - p_{1,t}$  or  $1 - p_{2,t}$ ). We here assume the absence of state assignment errors; thus, an individual in state  $s$  can only be encountered in state  $s$  (or be not encountered), but not encountered in another state. In other words, we allow only false-negative, but no false-positive errors. Once dead, an individual can no longer be encountered. This completes our mostly verbal description of the observation process, which is conditional on the state process, and thus the model is conceptually hierarchical. The model is conditional on first capture, that is, we do not model and estimate the initial capture probability. In Fig. 9.1, the two processes are shown graphically for an example.



**FIGURE 9.1** Example of the state and observation process of a marked individual over time for the multistate model. The sequence of true states in this individual is  $z = [1, 2, 2, 1, 1, 3, 3]$ , and the observed capture-history is  $y = [1, 2, 0, 1, 0, 0, 0]$ .

For an algebraic description of our two-location model, assume matrix  $\mathbf{z}$  with element  $z_{i,t}$ , which indicates the true state of individual  $i$  at time  $t$ . States are numbered from 1 to  $S$ ;  $S$  is the number of true states. Moreover, let us assume that vector  $f_{S_i}$  denotes the true state of individual  $i$  at its first encounter. As we assume there are no state assignment errors,  $f_{S_i}$  is identical to the observed state at first encounter. Furthermore, we define the four-dimensional state-transition matrix ( $\Omega$ ). The first and second dimension of  $\Omega$  denote the states of departure and of arrival, respectively, the third dimension the individual ( $i$ ), and the fourth dimension time ( $t$ ). Element  $\omega_{n,m,i,t}$  of  $\Omega$  is the probability that individual  $i$ , which is in state  $n$  at time  $t$ , is in state  $m$  at  $t+1$ . The observation matrix ( $\Theta$ ) also has four dimensions, the first denotes the true state of an individual, the second the observed state, and the remaining two dimensions are for individual and time. The element  $\varphi_{n,m,i,t}$  of  $\Theta$  is the probability that individual  $i$ , which is in state  $n$  at time  $t$ , is observed in state  $m$  at time  $t$ . The first two dimensions of the state-transition matrix  $\Omega$  are identical ( $S \times S$ ), but this need not be the case for the observation matrix ( $\Theta$ ). The first dimension of  $\Theta$  must also be  $S$  (the true number of states), but the second dimension is  $O$  (the number of observed states) which can be smaller or larger than  $S$ . As usual, the state-space model then consists of two model parts. The set of state equations is

$$\begin{aligned} z_{i,f_i} &= f_{S_i} \\ z_{i,t+1} | z_{i,t} &\sim \text{categorical}(\Omega_{z_{i,t}, 1 \dots S_i, t}). \end{aligned}$$

The first equation describes the state at the first encounter that is assumed to be assigned without error. Note that the argument of the categorical distribution is a vector of length  $S$ , that is, it is the complete row of the matrix  $\Omega$  for given values of the first ( $z_{i,t}$ ), third ( $i$ ), and fourth ( $t$ ) dimension. The second equation describes the development of the state membership over time. The observation equation links the true state with the observed state:

$$y_{i,t} | z_{i,t} \sim \text{categorical}(\Theta_{z_{i,t}, 1 \dots O_i, t})$$

where  $y$  is the observed multistate capture–recapture data. Similarly as before, the argument of the categorical distribution is a vector of length  $O$ , that is, it is the complete row of the matrix  $\Theta$  for given values of the first ( $z_{i,t}$ ), third ( $i$ ), and fourth ( $t$ ) dimension. The state and the observation process are both defined for  $t \geq f_i$ .

The multistate capture–recapture model makes similar assumptions as the CJS model (Chapter 7): the transition and observation probabilities must be the same for all individuals at a given occasion and state and individuals must be independent of each other. Individuals and states are recorded without error, and no marks must be lost. Of course, it is possible

to relax some of these assumptions by adopting a more general model. For example, by incorporating age or group effects, the assumption of equal transition and observation probabilities is no longer required for all individuals, but only for the individuals within an age class or group. Some of these assumptions can be tested with appropriate goodness-of-fit tests (Pradel et al., 2003; Choquet et al., 2009).

Because multistate capture–recapture models are so extremely flexible, there is a host of possible models that we might present. We have selected a few models that we think represent frequently encountered problems. In our presentation, we first give a description of the model by showing the first two dimensions of the state-transition ( $\Omega$ ) and observation ( $\Theta$ ) matrices. The formulation of these matrices requires the definition of exhaustive sets of true and observed states. Typically, state-transition probabilities may be composed of several parameters that we want to estimate separately, for example, state-specific survival and movement probabilities. These matrices form the heart of multistate modeling. They are essential for your understanding of this model class in principle and for the way in which these models are written in the BUGS language in particular. After presentation of the matrices, we either simulate data or use a real-data example and fit the model. Throughout, we denote state-transition matrices with rows representing states at occasion  $t$  and column states at occasion  $t + 1$ . Sometimes states at time  $t$  (rows) are also called “states of departure”, whereas states at time  $t + 1$  (columns) are named “states of arrival”. For the observation matrix, true states are in rows and the observed states are in columns. In the list of true states, we always include the state “dead” and in the list of observed states the state “not seen/encountered”. This is because both matrices need to be row stochastic, that is, the probabilities in a row must sum to 1. In Appendix 2, we show two additional useful multistate models along with the BUGS code for their analysis.

## 9.2 ESTIMATION OF MOVEMENT BETWEEN TWO SITES

---

### 9.2.1 Model Description

This is one of the simplest multistate models. Consider individuals that are marked and recaptured at two sites (A and B) and that there is movement between sites. Recapture is not perfect, and we want to estimate site-specific survival, as well as movement probabilities between sites. This is a common problem, and for this very situation, multistate capture–recapture models were originally developed (Arnason, 1972; Hestbeck et al., 1991; Brownie et al., 1993).

We start by defining the three true states: “alive at site A”, “alive at site B”, and “dead”. Here is the state-transition matrix:

		True State at Time $t + 1$		
		Site A	Site B	Dead
True State at Time $t$	Site A	$\phi_A(1 - \psi_{AB})$	$\phi_A\psi_{AB}$	$1 - \phi_A$
	Site B	$\phi_B\psi_{BA}$	$\phi_B(1 - \psi_{BA})$	$1 - \phi_B$
	Dead	0	0	1

or simply

$$\begin{matrix} & \text{site A} & \text{site B} & \text{dead} \\ \text{site A} & \left[ \begin{matrix} \phi_A(1 - \psi_{AB}) & \phi_A\psi_{AB} & 1 - \phi_A \end{matrix} \right] \\ \text{site B} & \left[ \begin{matrix} \phi_B\psi_{BA} & \phi_B(1 - \psi_{BA}) & 1 - \phi_B \end{matrix} \right] \\ \text{dead} & \left[ \begin{matrix} 0 & 0 & 1 \end{matrix} \right] \end{matrix}$$

States are ordered from top to bottom and from left to right in the order as given above (“site A”, “site B”, and “dead”). For example, the probability of being in state “site B” at time  $t + 1$ , given presence in state “site A” at time  $t$ , is  $\phi_A\psi_{AB}$ . The parameters in this matrix are the site-specific survival probabilities ( $\phi_A$ ,  $\phi_B$ ) and the movement probabilities ( $\psi_{AB}$ ,  $\psi_{BA}$ ).

When written in this way, we make a crucial assumption: survival from  $t$  to  $t + 1$  refers to the state in which the individual is at time  $t$ , and there is no mortality during movement. Thus,  $\psi_{AB}$  is the probability that an individual, which survived at site A from time  $t$  to  $t + 1$ , moves to site B shortly before  $t + 1$ . This is the typical way how the survival and movement parameters are defined. If we wish, we could also define that movement comes first and then the individual survives with the site-specific survival probability of the new site, in which case the transition matrix is this:

$$\begin{matrix} & \text{site A} & \text{site B} & \text{dead} \\ \text{site A} & \left[ \begin{matrix} (1 - \psi_{AB})\phi_A & \psi_{AB}\phi_B & 1 - \sum \text{row}_1 \end{matrix} \right] \\ \text{site B} & \left[ \begin{matrix} \psi_{BA}\phi_A & (1 - \psi_{BA})\phi_B & 1 - \sum \text{row}_2 \end{matrix} \right] \\ \text{dead} & \left[ \begin{matrix} 0 & 0 & 1 \end{matrix} \right] \end{matrix}.$$

Joe and Pollock (2002) developed a general model in which the time of movement is a random variable with a known distribution.

Regardless of how the state matrix is defined, the list of observed states includes “seen in A”, “seen in B”, and “not seen”. The observation matrix links the true states with the observed states:

		Observation at Time $t$		
		Seen in A	Seen in B	Not seen
True State at Time $t$	Site A	$p_A$	0	$1 - p_A$
	Site B	0	$p_B$	$1 - p_B$
	Dead	0	0	1

or simply

$$\begin{matrix} & \text{seen in A} & \text{seen in B} & \text{not seen} \\ \text{site A} & \begin{bmatrix} p_A & 0 & 1 - p_A \end{bmatrix} \\ \text{site B} & \begin{bmatrix} 0 & p_B & 1 - p_B \end{bmatrix} \\ \text{dead} & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

The parameters in this matrix are the site-specific recapture probabilities ( $p_A$ ,  $p_B$ ). The true states at time  $t$  correspond to the rows of the observation matrix in the order “site A”, “site B”, and “dead” from top to bottom, and the observed states are in columns in the order “seen in A”, “seen in B”, and “not seen” from left to right. For example, the probability that an individual in true state “site A” at time  $t$  is in the observed state “not seen” at time  $t$  is  $1 - p_A$ .

For ease of presentation, we have dropped several indices in these matrices. In reality, however, both matrices could have an index  $t$  for time and  $i$  for individual.

### 9.2.2 Generation of Simulated Data

Imagine that we sampled capture–recapture data of little ringed plovers (Fig. 9.2) at two sites (A and B). Habitat quality at site A is higher than at site B, and thus annual survival in A is higher (0.8) than that in B (0.7), and movements from A to B are less likely ( $\psi_{AB} = 0.3$ ) than movements from B to A ( $\psi_{BA} = 0.5$ ). We assume that capture effort was higher at A than at B. In the data sets, captures of an individual at site A are labeled “1” and captures at site B “2”.

To generate multistate capture–recapture data, we first need to define the probabilities in both the state-transition and the observation matrix. For individual  $i$  released at time  $t$  in state  $m$ , we simulate its state at time  $t + 1$  using a categorical distribution. The probabilities of the categorical distribution are taken from row  $m$  of the state-transition matrix



**FIGURE 9.2** Little ringed plover (*Charadrius dubius*) (Photograph by D. Occhiato).

(`PSI.STATE`) of individual  $i$  at time  $t$ . We then simulate the observation for individual  $i$  at time  $t + 1$ . Assume that individual  $i$  is in state  $n$  at time  $t + 1$ . We then simulate the observation using a categorical distribution again. The probabilities of the categorical distribution are taken this time from row  $n$  of the observation matrix (`PSI.OBS`) for individual  $i$  at time  $t + 1$ . These steps are first repeated for individual  $i$  until the end of the study and then for all released individuals. In program R, we specify the categorical distribution as a multinomial distribution with trial size equal to 1. In the following code, we first define the simulation parameters, the state-transition and observation matrix and then apply the function `simul.ms`, which simulates multistate capture–recapture data as described earlier.

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phiA <- 0.8
phiB <- 0.7
psiAB <- 0.3
psibA <- 0.5
pA <- 0.7
pB <- 0.4
n.occasions <- 6
n.states <- 3
```

```

n.obs <- 3
marked <- matrix(NA, ncol = n.states, nrow = n.occasions)
marked[, 1] <- rep(100, n.occasions)
marked[, 2] <- rep(60, n.occasions)
marked[, 3] <- rep(0, n.occasions)

# Define matrices with survival, transition and recapture probabilities
# These are 4-dimensional matrices, with
# Dimension 1: state of departure
# Dimension 2: state of arrival
# Dimension 3: individual
# Dimension 4: time

# 1. State process matrix
totrel <- sum(marked) * (n.occasions - 1)
PSI.STATE <- array(NA, dim = c(n.states, n.states, totrel, n.occasions - 1))
for (i in 1:totrel) {
  for (t in 1:(n.occasions - 1)) {
    PSI.STATE[,,i,t] <- matrix(c(
      phiA * (1 - psiAB), phiA * psiAB, 1 - phiA,
      phiB * psiBA, phiB * (1 - psiBA), 1 - phiB,
      0, 0, 1), nrow = n.states,
      byrow = TRUE)
  } #t
} #i

# 2. Observation process matrix
PSI.OBS <- array(NA, dim = c(n.states, n.obs, totrel, n.occasions - 1))
for (i in 1:totrel) {
  for (t in 1:(n.occasions - 1)) {
    PSI.OBS[,,i,t] <- matrix(c(
      pA, 0, 1 - pA,
      0, pB, 1 - pB,
      0, 0, 1), nrow = n.states, byrow = TRUE)
  } #t
} #i

# Define function to simulate multistate capture-recapture data
simul.ms <- function(PSI.STATE, PSI.OBS, marked, unobservable = NA) {
  # Unobservable: number of state that is unobservable
  n.occasions <- dim(PSI.STATE)[4] + 1
  CH <- CH.TRUE <- matrix(NA, ncol = n.occasions, nrow = sum(marked))
  # Define a vector with the occasion of marking
  mark.occ <- matrix(0, ncol = dim(PSI.STATE)[1], nrow = sum(marked))
  g <- colSums(marked)
  for (s in 1:dim(PSI.STATE)[1]) {
    if (g[s] == 0) next # To avoid error message if nothing to replace
    mark.occ[(cumsum(g[1:s]) - g[s] + 1)[s]:cumsum(g[1:s])[s], s] <-
      rep(1:n.occasions, marked[1:n.occasions, s])
  } #s
  for (i in 1:sum(marked)) {
    for (s in 1:dim(PSI.STATE)[1]) {
      if (mark.occ[i, s] == 0) next
      first <- mark.occ[i, s]
    }
  }
}

```

```

CH[i,first] <- s
CH.TRUE[i,first] <- s
} #s
for (t in (first+1):n.occasions){
  # Multinomial trials for state transitions
  if (first==n.occasions) next
  state <- which(rmultinom(1, 1, PSI.STATE[CH.TRUE[i,t-1],,
    i,t-1]) == 1)
  CH.TRUE[i,t] <- state
  # Multinomial trials for observation process
  event <- which(rmultinom(1, 1, PSI.OBS[CH.TRUE[i,t],,i,
    t-1]) == 1)
  CH[i,t] <- event
} #t
} #i
# Replace the NA and the highest state number (dead) in the file by 0
CH[is.na(CH)] <- 0
CH[CH==dim(PSI.STATE)[1]] <- 0
CH[CH==unobservable] <- 0
id <- numeric(0)
for (i in 1:dim(CH)[1]){
  z <- min(which(CH[i,] != 0))
  ifelse(z==dim(CH)[2], id <- c(id,i), id <- c(id))
}
return(list(CH=CH[-id,], CH.TRUE=CH.TRUE[-id,]))
# CH: capture-histories to be used
# CH.TRUE: capture-histories with perfect observation
}

# Execute function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

```

The multistate capture–recapture data simulated may look like this (first five rows shown):

[1,]	1	2	0	0	0	0
[2,]	1	0	0	0	0	0
[3,]	1	0	1	1	0	0
[4,]	1	0	2	0	0	0
[5,]	1	0	2	0	0	0

A “1” denotes capture at site A, a “2” capture at site B, and a “0” denotes noncapture. To fit a multistate model in the state-space formulation, we need to compute a vector indicating the occasion of marking for each individual. Furthermore, we must replace the “0” in the data with a “3” to match the observed states, which are numbered “1” (seen at site A), “2” (seen at site B), and “3” (not seen).

```

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

```

```
# Recode CH matrix: note, a 0 is not allowed in WinBUGS!
# 1 = seen alive in A, 2 = seen alive in B, 3 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 3
```

The final multistate capture–recapture data ready to be analyzed in BUGS is named rCH and may look like this:

[1,]	1	2	3	3	3	3
[2,]	1	3	3	3	3	3
[3,]	1	3	1	1	3	3
[4,]	1	3	2	3	3	3
[5,]	1	3	2	3	3	3

### 9.2.3 Analysis of the Model

As so often, the BUGS code to analyze the multistate capture–recapture data closely resembles the way in which we simulate the data. In addition, we define priors for each parameter and can impose constraints on them. Then, we define the state-transition and the observation matrices using the model parameters. They are named  $\text{ps}$  and  $\text{po}$ , respectively, in the BUGS code, and correspond to  $\Omega$  and  $\Theta$  of Section 9.1. These matrices have four dimensions because they not only indicate states of departure and arrival but also time and individual. Finally, the state-space likelihood section of code is similar to that introduced for CJS models. The state-transition process describes the probability of the state at time  $t + 1$  (stored in matrix  $Z$ ) conditional on the state at time  $t$ , and the observation process describes the mapping of the state at time  $t + 1$  on the observed data. The only difference with the CJS model is that in the likelihood we now use a categorical instead of a Bernoulli distribution. Note that the categorical distribution in WinBUGS needs the empty index in the last position; hence, the matrix dimensions are ordered differently from the formulas in Section 9.1.

```
# Specify model in BUGS language
sink("ms.bug")
cat("
model {

# -----
# Parameters:
# phiA: survival probability at site A
# phiB: survival probability at site B
# psiAB: movement probability from site A to site B
# psiBA: movement probability from site B to site A
# pA: recapture probability at site A
# pB: recapture probability at site B
# -----
# States (S):
# 1 alive at A
```

```

# 2 alive at B
# 3 dead
# Observations (O):
# 1 seen at A
# 2 seen at B
# 3 not seen
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)){
  phiA[t] <- mean.phi[1]
  phiB[t] <- mean.phi[2]
  psiAB[t] <- mean.psi[1]
  psiBA[t] <- mean.psi[2]
  pA[t] <- mean.p[1]
  pB[t] <- mean.p[2]
}
for (u in 1:2){
  mean.phi[u] ~ dunif(0, 1)    # Priors for mean state-spec. survival
  mean.psi[u] ~ dunif(0, 1)    # Priors for mean transitions
  mean.p[u] ~ dunif(0, 1)      # Priors for mean state-spec. recapture
}

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- phiA[t] * (1-psiAB[t])
    ps[1,i,t,2] <- phiA[t] * psiAB[t]
    ps[1,i,t,3] <- 1-phiA[t]
    ps[2,i,t,1] <- phiB[t] * psiBA[t]
    ps[2,i,t,2] <- phiB[t] * (1-psiBA[t])
    ps[2,i,t,3] <- 1-phiB[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 1
    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- pA[t]
    po[1,i,t,2] <- 0
    po[1,i,t,3] <- 1-pA[t]
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- pB[t]
    po[2,i,t,3] <- 1-pB[t]
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 0
    po[3,i,t,3] <- 1
  } #t
} #i

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){

```

```

# State process: draw S(t) given S(t-1)
z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
# Observation process: draw O(t) given S(t)
y[i,t] ~ dcat(po[z[i,t], i, t-1,])
} #t
} #i
",fill = TRUE)
sink()

```

Before we load the data, we may again compute a matrix with known latent states  $z$ . This will reduce computation time and improve convergence; see Section 7.3.1. The latent state  $z$  now contains information not only about survival as in the CJS or the mark-recovery model but also about a location (or “state”). Therefore, the latent state is always unknown during occasions when an individual is not encountered. Care must also be taken when computing the known-state matrix when the names of the true and the observed states differ. The following function creates the required matrix for the current example.

```

# Function to create known latent states z
known.state.ms <- function(ms, notseen) {
  # notseen: label for 'not seen'
  state <- ms
  state[state==notseen] <- NA
  for (i in 1:dim(ms)[1]) {
    m <- min(which(!is.na(state[i])))
    state[i,m] <- NA
  }
  return(state)
}

```

Initial values need to be given only for the unknown  $z$ ; the following function creates these values.

```

# Function to create initial values for unknown z
ms.init.z <- function(ch, f) {
  for (i in 1:dim(ch)[1]) {ch[i,1:f[i]] <- NA}
  states <- max(ch, na.rm=TRUE)
  known.states <- 1:(states-1)
  v <- which(ch==states)
  ch[-v] <- NA
  ch[v] <- sample(known.states, length(v), replace=TRUE)
  return(ch)
}

```

Now, we are ready to fit the model.

```

# Bundle data
bugs.data <- list(y = rCH, f = f, n.occasions = dim(rCH)[2], nind =
  dim(rCH)[1], z = known.state.ms(rCH, 3))

```

```

# Initial values
inits <- function(){list(mean.phi = runif(2, 0, 1), mean.psi = runif(2,
  0, 1), mean.p = runif(2, 0, 1), z=ms.init.z(rCH, f))}

# Parameters monitored
parameters <- c("mean.phi", "mean.psi", "mean.p")

# MCMC settings
ni <- 10000
nt <- 6
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 8 min)
ms <- bugs(bugs.data, inits, parameters, "ms.bug", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory=
  bugs.dir, working.directory=getwd())

```

This model converges relatively quickly.

```
print(ms, digits = 3)
```

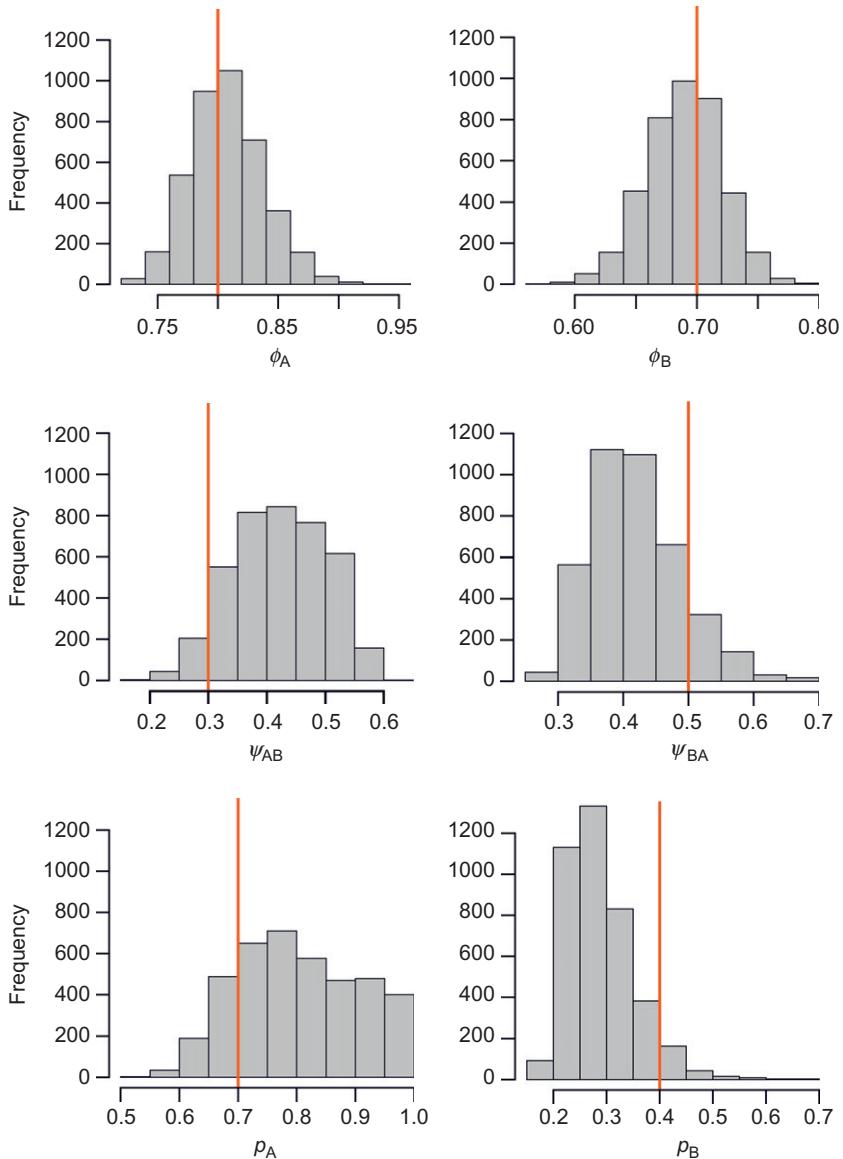
	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
mean.phi[1]	0.808	0.030	0.753	0.786	0.806	0.826	0.872	1.005	550
mean.phi[2]	0.690	0.031	0.627	0.670	0.691	0.711	0.748	1.001	4000
mean.psi[1]	0.422	0.079	0.271	0.363	0.423	0.485	0.559	1.007	480
mean.psi[2]	0.419	0.069	0.311	0.367	0.412	0.460	0.578	1.005	660
mean.p[1]	0.803	0.102	0.626	0.723	0.795	0.888	0.985	1.009	310
mean.p[2]	0.288	0.062	0.201	0.242	0.277	0.322	0.430	1.008	590

The posterior distributions of the six model parameters look reasonably good (Fig. 9.3). However, the posteriors for transitions and recapture parameters are wide, indicating that these parameters are not estimated precisely. This is probably because the study duration was short in our example, so the data contained little information about these parameters.

```

par(mfrow = c(3, 2), las = 1)
hist(ms$sims.list$mean.phi[,1], col = "gray", main = "", xlab =
  expression(phi[A]), ylim=c(0,1300))
abline(v=phiA, col = "red")
hist(ms$sims.list$mean.phi[,2], col = "gray", main = "", xlab =
  expression(phi[B]), ylim=c(0,1300), ylab="")
abline(v=phiB, col="red")
hist(ms$sims.list$mean.psi[,1], col = "gray", main = "", xlab =
  expression(psi[AB]), ylim=c(0,1300))
abline(v=psiAB, col="red")
hist(ms$sims.list$mean.psi[,2], col = "gray", main = "", xlab =
  expression(psi[BA]), ylab="", ylim=c(0,1300))
abline(v=psiBA, col="red")
hist(ms$sims.list$mean.p[,1], col = "gray", main = "", xlab =
  expression(p[A]), ylim=c(0,1300))
abline(v=pA, col = "red")
hist(ms$sims.list$mean.p[,2], col = "gray", main = "", xlab =
  expression(p[B]), ylab="", ylim=c(0,1300))
abline(v=pB, col = "red")

```



**FIGURE 9.3** Posterior distributions of survival, movement, and recapture probabilities. Red lines give the values of the data generating parameters.

This basic model can now be extended in the same way as we saw for the CJS model in Chapter 7. Thus, it is straightforward to add fixed time effects, group effects, or to include individual or temporal random effects. As mentioned earlier, all we need to do is to modify the code sections called “Priors and

constraints" and "Define parameters". You may want to check the examples given for the CJS models to understand how to make these changes and study exercises 1 and 2 in Section 9.9.

The same model structure may be used when states A and B denote disease states, classes of reproductive success, or "breeder" and "non-breeder", to name a few possibilities. Moreover, this model easily extends to a larger number of locations (see Section 9.5).

This multistate model is written in such a way that the inclusion of temporal and individual effects is straightforward, that is, only the model section called "Priors and constraints" needs to be modified. However, without individual effects, the model could be written in a more efficient way, which would reduce computing time somewhat. In particular, the definition of the state-transition and observation matrices ( $\text{ps}$  and  $\text{po}$ ) does not need the index for individual in that case. Thus, an alternative way to write the model for this case is as follows:

```
# Specify model in BUGS language
sink("ms.alternative1.bug")
cat("
model {
  # Priors and constraints
  for (t in 1:(n.occasions-1)){
    phiA[t] <- mean.phi [1]
    phiB[t] <- mean.phi [2]
    psiAB[t] <- mean.psi [1]
    psiBA[t] <- mean.psi [2]
    pA[t] <- mean.p[1]
    pB[t] <- mean.p[2]
  }
  for (u in 1:2){
    mean.phi[u] ~ dunif(0, 1)      # Priors for mean state-spec. survival
    mean.psi[u] ~ dunif(0, 1)      # Priors for mean transitions
    mean.p[u] ~ dunif(0, 1)        # Priors for mean state-spec. recapture
  }
  # Define state-transition and observation matrices
  # Define probabilities of state S(t+1) given S(t)
  for (t in 1:(n.occasions-1)){
    ps[1,t,1] <- phiA[t] * (1-psiAB[t])
    ps[1,t,2] <- phiA[t] * psiAB[t]
    ps[1,t,3] <- 1-phiA[t]
    ps[2,t,1] <- phiB[t] * psiBA[t]
    ps[2,t,2] <- phiB[t] * (1-psiBA[t])
    ps[2,t,3] <- 1-phiB[t]
    ps[3,t,1] <- 0
    ps[3,t,2] <- 0
    ps[3,t,3] <- 1
  }
  # Define probabilities of O(t) given S(t)
  po[1,t,1] <- pA[t]
  po[1,t,2] <- 0
}
```

```

po[1,t,3] <- 1-pA[t]
po[2,t,1] <- 0
po[2,t,2] <- pB[t]
po[2,t,3] <- 1-pB[t]
po[3,t,1] <- 0
po[3,t,2] <- 0
po[3,t,3] <- 1
} #t

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions) {
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], t-1,])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], t-1,])
  } #t
} #i
",fill=TRUE)
sink()

```

If in addition we do not need temporal effects, there is an even more efficient way to write the model: the index for time in the matrices `ps` and `po` is no longer needed. We then define priors directly for the quantities of interest (`phiA`, `phiB`, ...), and we also need to change the initial values accordingly.

```

# Specify model in BUGS language
sink("ms.alternative2.bug")
cat("
model {

# Priors and constraints
phiA ~ dunif(0, 1)      # Prior for mean survival in A
phiB ~ dunif(0, 1)      # Prior for mean survival in B
psiAB ~ dunif(0, 1)      # Prior for mean movement from A to B
psiBA ~ dunif(0, 1)      # Prior for mean movement from B to A
pA ~ dunif(0, 1)         # Prior for mean recapture in A
pB ~ dunif(0, 1)         # Prior for mean recapture in B

# Define state-transition and observation matrices
# Define probabilities of state S(t+1) given S(t)
ps[1,1] <- phiA * (1-psiAB)
ps[1,2] <- phiA * psiAB
ps[1,3] <- 1-phiA
ps[2,1] <- phiB * psibA
ps[2,2] <- phiB * (1-psibA)
ps[2,3] <- 1-phiB
ps[3,1] <- 0
ps[3,2] <- 0
ps[3,3] <- 1

```

```

# Define probabilities of O(t) given S(t)
po[1,1] <- pA
po[1,2] <- 0
po[1,3] <- 1-pA
po[2,1] <- 0
po[2,2] <- pB
po[2,3] <- 1-pB
po[3,1] <- 0
po[3,2] <- 0
po[3,3] <- 1

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions) {
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1],])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t],])
    } #t
  } #i
}
",fill = TRUE)
sink()

```

The reduction in computing time is about 30% for the first and about 40% for the second alternative model. Despite these computational benefits, we will not use these parameterizations in the following examples but stick to the more general parameterization. However, we recommend using alternative parameterizations when computing time is an issue.

## 9.3 ACCOUNTING FOR TEMPORARY EMIGRATION

### 9.3.1 Model Description

Sometimes individuals are not available for capture at certain occasions, although they are alive *somewhere*. An example of this is when individuals skip reproduction in a particular year, and sampling is conducted only at breeding sites, or plants that are dormant in some years when they are only surveyed above ground. If the temporary absence is random (i.e., does not depend on whether an individual was available for capture at the previous occasion), then temporary emigration does not bias estimates of survival probability. It does lower recapture probability, but this parameter is usually not of direct interest. By contrast, if temporary absence is Markovian, that is, depends on whether an individual was absent one time step ago, then it does cause bias in the estimated survival probability, unless accounted for (Kendall et al., 1997; Schaub

et al., 2004a). Note that temporary emigration is different from permanent emigration. When individuals emigrate permanently, they become permanently unavailable for capture and the estimated survival probability is always biased low. That is why all survival estimated from CJS and multi-state models should more accurately be called “apparent survival” (see Chapter 7).

To model temporary emigration, we define the states “alive and present”, “alive and absent”, and “dead”. The state-transition matrix is

$$\begin{array}{ccc} & \text{present} & \text{absent} & \text{dead} \\ \text{present} & \left[ \begin{array}{ccc} \phi(1 - \psi_{IO}) & \phi\psi_{IO} & 1 - \phi \end{array} \right], \\ \text{absent} & \left[ \begin{array}{ccc} \phi\psi_{OI} & \phi(1 - \psi_{OI}) & 1 - \phi \end{array} \right], \\ \text{dead} & \left[ \begin{array}{ccc} 0 & 0 & 1 \end{array} \right] \end{array}$$

where  $\phi$  is survival probability,  $\psi_{IO}$  is the probability that an individual present (“in”) at time  $t$  is absent (“out”) at time  $t + 1$ , and  $\psi_{OI}$  is the probability that an individual absent at time  $t$  is present at time  $t + 1$ . If  $\psi_{IO} = 1 - \psi_{OI}$ , temporary emigration is random.

The list of possible observations is “seen” and “not seen”. Therefore, we have a  $3 \times 2$  observation matrix,

$$\begin{array}{cc} & \text{seen} & \text{not seen} \\ \text{present} & \left[ \begin{array}{cc} p & 1 - p \end{array} \right], \\ \text{absent} & \left[ \begin{array}{cc} 0 & 1 \end{array} \right], \\ \text{dead} & \left[ \begin{array}{cc} 0 & 1 \end{array} \right] \end{array}$$

where  $p$  is recapture probability (conditional on being alive and available for capture).

The model of temporary emigration is an example of a model with two unobserved states (“alive and absent” and “dead”). Such models often have identifiability problems. Kendall and Nichols (2002), Schaub et al. (2004a), and Bailey et al. (2010) studied the identifiability and the performance of this and more complex models with additional states, and give catalogues of which parameters can be estimated under which conditions.

### 9.3.2 Generation of Simulated Data

Let us assume that we have repeatedly photographed fire salamanders (Fig. 9.4) at a hibernation site in a cave. The black and yellow pattern is extremely variable among individuals; hence, these photos allow one to determine individual identity. Annual adult survival of fire salamanders is high (Schmidt et al., 2005); we here assume  $\phi = 0.85$ . Since salamanders do not hibernate every year at the same place, they may temporarily be absent, that is, unavailable for capture (which means, being



**FIGURE 9.4** Fire salamander (*Salamandra salamandra*), Switzerland, 2008 (Photograph by T. Ott).

photographed), when sampling only takes place at one cave. We assume that the probability an individual that is present becomes absent in the next year is  $\psi_{IO} = 0.2$ , and that the probability a salamander that is absent becomes present the next year is  $\psi_{OI} = 0.3$ . Given presence in the cave, recapture probability is  $p = 0.7$ .

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phi <- 0.85
psiIO <- 0.2
psiOI <- 0.3
p <- 0.7
n.occasions <- 8
n.states <- 3
n.obs <- 2
marked <- matrix(NA, ncol=n.states, nrow=n.occasions)
marked[,1] <- rep(70, n.occasions) # Present
marked[,2] <- rep(0, n.occasions) # Absent
marked[,3] <- rep(0, n.occasions) # Dead
# Define matrices with survival, transition and recapture probabilities
# These are 4-dimensional matrices, with
# Dimension 1: state of departure
```

```

# Dimension 2: state of arrival
# Dimension 3: individual
# Dimension 4: time
# 1. State process matrix
totrel <- sum(marked) * (n.occasions-1)
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel,
    n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.STATE[,,i,t] <- matrix(c(
      phi*(1-psiIO), phi*psiIO, 1-phi,
      phi*psiOI, phi*(1-psiOI), 1-phi,
      0, 0, 1 ), nrow=n.states,
      byrow=TRUE)
  } #t
} #i

# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.OBS[,,i,t] <- matrix(c(
      p, 1-p,
      0, 1,
      0, 1 ), nrow=n.states, byrow=TRUE)
  } #t
} #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed!
# 1 = seen alive, 2 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 2

```

The elements of the capture-history matrix are equal to “1” at occasions when an individual was seen and “2” otherwise. So far, we always removed the zeroes in the capture-histories; however, this would not really be necessary for zeroes that occur before first capture. Remember that in the CJS, mark-recovery, and multistate models of Chapters 7–9, the data before first capture are not modeled, that is, they do not enter the likelihood.

### 9.3.3 Analysis of the Model

Again, the BUGS code is straightforward to write in terms of the state-transition and observation matrices. Both matrices are specified along with the priors; no change is required in the remaining code.

```

# Specify model in BUGS language
sink("tempemi.bug")
cat("
model {

# -----
# Parameters:
# phi: survival probability
# psiIO: probability to emigrate
# psioI: probability to immigrate
# p: recapture probability
# -----
# States (S):
# 1 alive and present
# 2 alive and absent
# 3 dead
# Observations (O):
# 1 seen
# 2 not seen
# -----


# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi[t] <- mean.phi
  psiIO[t] <- mean.psiIO
  psioI[t] <- mean.psioI
  p[t] <- mean.p
}
mean.phi ~ dunif(0, 1)      # Prior for mean survival
mean.psiIO ~ dunif(0, 1)    # Prior for mean temp. emigration
mean.psioI ~ dunif(0, 1)    # Prior for mean temp. immigration
mean.p ~ dunif(0, 1)        # Prior for mean recapture

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- phi[t] * (1-psioI[t])
    ps[1,i,t,2] <- phi[t] * psiIO[t]
    ps[1,i,t,3] <- 1-phi[t]
    ps[2,i,t,1] <- phi[t] * psioI[t]
    ps[2,i,t,2] <- phi[t] * (1-psioI[t])
    ps[2,i,t,3] <- 1-phi[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- p[t]
    po[1,i,t,2] <- 1-p[t]
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- 1
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 1
  } #t
} #i
}

```

```

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], i, t-1,])
  } #t
} #i
}, fill = TRUE)
sink()

# Bundle data
bugs.data <- list(y = rCH, f = f, n.occasions = dim(rCH) [2],
  nind = dim(rCH) [1], z = known.state.ms(rCH, 2))

# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1), mean.psiIO = runif(1,
  0, 1), mean.psiOI = runif(1, 0, 1), mean.p = runif(1, 0, 1), z = ms.init.z
  (rCH, f))}

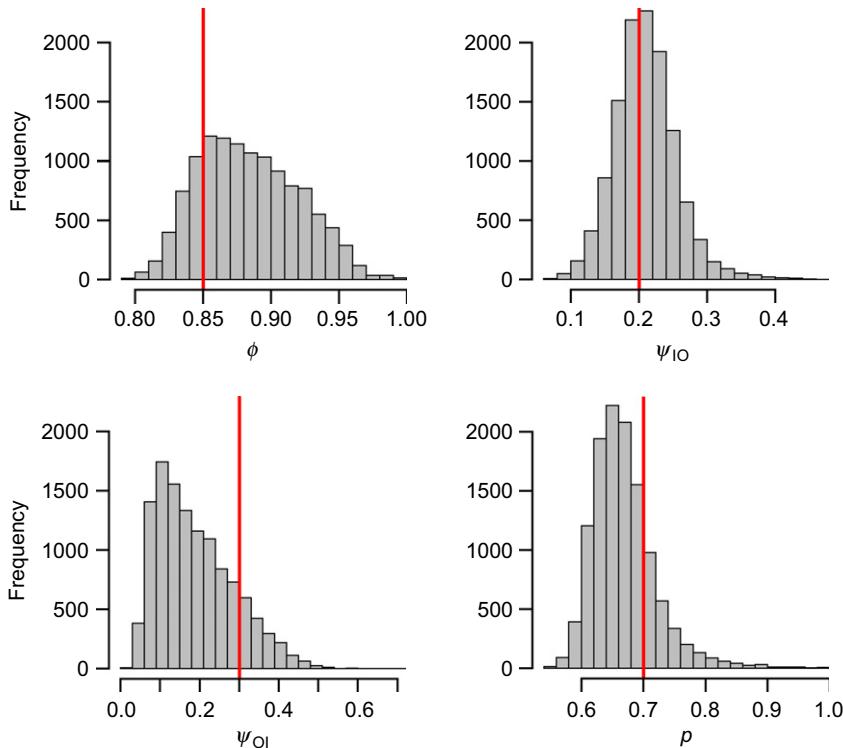
# Parameters monitored
parameters <- c("mean.phi", "mean.psiIO", "mean.psiOI", "mean.p")

# MCMC settings
ni <- 50000
nt <- 10
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 35 min)
tempemi <- bugs(bugs.data, inits, parameters, "tempemi.bug", n.chains =
  nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

```

The analysis of this simulated data set requires long Markov chains to achieve convergence. This model has identifiability problems, the degree of which depends on the parameter values. Recapture and transition probabilities are not identifiable and biased if temporary emigration is random (i.e.,  $\psi_{IO} = 1 - \psi_{OI}$ ); see Kendall and Nichols (2002) and Schaub et al. (2004a). In our case, however, the posterior distributions of all parameters match up well with the values used for generating the data set, with the exception of the probability of moving back to the study area (temporary immigration; Fig. 9.5). Again, the discrepancy stems from the fact that the simulated data set is relatively small. Repeated simulations would show that the estimators are unbiased, but that their precision is low (Schaub et al., 2004a). Low precision of transition parameters is typical for multistate models with unobservable states.



**FIGURE 9.5** Posterior distributions of the estimated parameters along with the values used to simulate the data (red vertical lines).

```

print(tempemi, digits = 3)

      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
mean.phi  0.884  0.037  0.823  0.855  0.880  0.910  0.956  1.011    260
mean.psiIO 0.210  0.046  0.126  0.180  0.207  0.235  0.310  1.004    610
mean.psiOI 0.189  0.096  0.057  0.111  0.170  0.252  0.407  1.008    370
mean.p     0.669  0.052  0.592  0.634  0.661  0.693  0.796  1.003  12000

par(mfrow = c(2, 2), las = 1)
hist(tempemi$sims.list$mean.phi, col = "gray", main = "", xlab =
  expression(phi))
abline(v = phi, col = "red", lwd = 2)
hist(tempemi$sims.list$mean.psioI, col = "gray", main = "", xlab =
  expression(psi[IO]), ylab = "")
abline(v = psioI, col = "red", lwd = 2)
hist(tempemi$sims.list$mean.psiOI, col = "gray", main = "", xlab =
  expression(psi[OI]))
abline(v = psiOI, col = "red", lwd = 2)
hist(tempemi$sims.list$mean.p, col = "gray", main = "", xlab =
  expression(p), ylab = "")
abline(v = p, col = "red", lwd = 2)

```

## 9.4 ESTIMATION OF AGE-SPECIFIC PROBABILITY OF FIRST BREEDING

### 9.4.1 Model Description

For many long-lived species, there is an interest in estimating age-specific probability of first breeding (reproduction). This requires that newborn individuals are marked and later resighted/recaptured when they breed. The state associated with a resighting event is “seen, no breeder yet” if an individual is observed but is not breeding in a given year, nor has it ever been observed breeding before, or “seen as breeder” if the individual is seen breeding in a year. If an individual is resighted as a non-breeder in a year but was observed to breed earlier, it is also assigned to the state “seen as breeder”. (Note that this assumption could be relaxed by the inclusion of a further state allowing the estimation of breeding frequency of experienced breeders.) If such data are analyzed without accounting for imperfect detection, age at first breeding is overestimated because some individuals may have first bred *before* they were first *observed* breeding. The multistate capture–recapture model described here avoids this bias by accounting for imperfect detection.

The model makes the assumption that the age at which all previous non-breeders start to breed is known. In our example, we assume that all individuals 3 years old have become breeders and define the following states in the model: “juvenile”, “not yet breeding at age 1”, “not yet breeding at age 2”, “breeder”, and “dead”. The resulting state-transition matrix is

$$\begin{array}{ccccc} & \text{juvenile} & \text{non-breeder 1} & \text{non-breeder 2} & \text{breeder} & \text{dead} \\ \text{juvenile} & 0 & \phi_1(1 - \alpha_1) & 0 & \phi_1\alpha_1 & 1 - \phi_1 \\ \text{non-breeder 1} & 0 & 0 & \phi_2(1 - \alpha_2) & \phi_2\alpha_2 & 1 - \phi_2 \\ \text{non-breeder 2} & 0 & 0 & 0 & \phi_{ad} & 1 - \phi_{ad} \\ \text{breeder} & 0 & 0 & 0 & \phi_{ad} & 1 - \phi_{ad} \\ \text{dead} & 0 & 0 & 0 & 0 & 1 \end{array}.$$

Here,  $\phi$  denotes the age-specific survival probability and  $\alpha_x$  the probabilities of starting to breed at age  $x$ . We define three age classes for survival, but other definitions would also be possible.

The possible observations are “seen as juv”, “seen as non-breeder”, “seen as breeder”, and “not seen”. The observation matrix is

$$\begin{array}{ccccc} & \text{seen as juv} & \text{seen non-breeder} & \text{seen breeder} & \text{not seen} \\ \text{juvenile} & 0 & 0 & 0 & 1 \\ \text{non-breeder 1} & 0 & p_{NB} & 0 & 1 - p_{NB} \\ \text{non-breeder 2} & 0 & p_{NB} & 0 & 1 - p_{NB} \\ \text{breeder} & 0 & 0 & p_B & 1 - p_B \\ \text{dead} & 0 & 0 & 0 & 1 \end{array},$$

where  $p_{NB}$  is the probability of re-encountering an individual that is not yet breeding and  $p_B$  is the probability of re-encountering a breeder. Depending on the study,  $p_{NB}$  may be zero.

The parameters estimated in this model allow the estimation of the mean age at first reproduction or of age-specific recruitment probabilities. Pradel and Lebreton (1999) show the connections between these quantities.

### 9.4.2 Generation of Simulated Data

Let us assume that we studied survival and age-specific recruitment in little ringed plovers (Fig. 9.2). We color-mark chicks in a population and resight them in subsequent years. For all resighted birds, we record their breeding status. In the simulation, we assume the following parameters:  $\phi_1 = 0.4$ ,  $\phi_2 = 0.7$ ,  $\phi_{ad} = 0.8$ ,  $\alpha_1 = 0.2$ ,  $\alpha_2 = 0.6$ ,  $p_{NB} = 0.5$ , and  $p_B = 0.7$ .

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phi.1 <- 0.4
phi.2 <- 0.7
phi.ad <- 0.8
alpha.1 <- 0.2
alpha.2 <- 0.6
p.NB <- 0.5
p.B <- 0.7
n.occasions <- 7
n.states <- 5
n.obs <- 4
marked <- matrix(0, ncol=n.states, nrow=n.occasions)
marked[,1] <- rep(100, n.occasions) # Releases only as juveniles

# Define matrices with survival, transition and recapture probabilities
# These are 4-dimensional matrices, with
# Dimension 1: state of departure
# Dimension 2: state of arrival
# Dimension 3: individual
# Dimension 4: time
# 1. State process matrix
totrel <- sum(marked) * (n.occasions-1)
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.STATE[,,i,t] <- matrix(c(
      0, phi.1*(1-alpha.1), 0, phi.1*alpha.1, 1-phi.1,
      0, 0, phi.2*(1-alpha.2), phi.2*alpha.2, 1-phi.2,
      0, 0, 0, phi.ad, 1-phi.ad,
      0, 0, 0, phi.ad, 1-phi.ad,
      0, 0, 0, 0, 1), nrow=
      n.states, byrow=TRUE)
  }
}
} #i
```

```

# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel) {
  for (t in 1:(n.occasions-1)) {
    PSI.OBS[,,i,t] <- matrix(c(
      0, 0, 0, 1,
      0, p.NB, 0, 1-p.NB,
      0, p.NB, 0, 1-p.NB,
      0, 0, p.B, 1-p.B,
      0, 0, 0, 1), nrow = n.states, byrow = TRUE)
  } #t
} #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed!
# 1 = seen as juv, 2 = seen no rep, 3 = seen rep, 4 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 4

```

The capture-histories have a “1” when a chick was marked, a “2” when an as yet non-breeding individual is observed, and a “3” when an individual is either observed breeding or it is observed non-breeding but has been observed breeding already at previous occasions. Finally, at occasions when an individual is not observed, the capture-history contains a “4”.

#### 9.4.3 Analysis of the Model

Here is the BUGS code for the analysis of the model.

```

# Specify model in BUGS language
sink("agerecruitment.bug")
cat("
model {

# -----
# Parameters:
# phi.1: first year survival probability
# phi.2: second year survival probability
# phi.ad: adult survival probability
# alpha.1: probability to start breeding when 1 year old
# alpha.2: probability to start breeding when 2 years old
# p.NB: recapture probability of non-breeders
# p.B: recapture probability of breeders
# -----
# States (S):
# 1 juvenile
# 2 not yet breeding at age 1 year

```

```

# 3 not yet breeding at age 2 years
# 4 breeder
# 5 dead
# Observations (O):
# 1 seen as juvenile
# 2 seen as not yet breeding
# 3 seen breeding
# 4 not seen
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi.1[t] <- mean.phi1
  phi.2[t] <- mean.phi2
  phi.ad[t] <- mean.phiad
  alpha.1[t] <- mean.alphal1
  alpha.2[t] <- mean.alpha2
  p.NB[t] <- mean.pNB
  p.B[t] <- mean.pB
}
mean.phi1 ~ dunif(0, 1)      # Prior for mean 1y survival
mean.phi2 ~ dunif(0, 1)      # Prior for mean 2y survival
mean.phiad ~ dunif(0, 1)     # Prior for mean ad survival
mean.alphal1 ~ dunif(0, 1)   # Prior for mean 1y breeding prob.
mean.alpha2 ~ dunif(0, 1)    # Prior for mean 2y breeding prob.
mean.pNB ~ dunif(0, 1)      # Prior for mean recapture non-breeders
mean.pB ~ dunif(0, 1)        # Prior for mean recapture breeders

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- 0
    ps[1,i,t,2] <- phi.1[t] * (1-alpha.1[t])
    ps[1,i,t,3] <- 0
    ps[1,i,t,4] <- phi.1[t] * alpha.1[t]
    ps[1,i,t,5] <- 1-phi.1[t]
    ps[2,i,t,1] <- 0
    ps[2,i,t,2] <- 0
    ps[2,i,t,3] <- phi.2[t] * (1-alpha.2[t])
    ps[2,i,t,4] <- phi.2[t] * alpha.2[t]
    ps[2,i,t,5] <- 1-phi.2[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 0
    ps[3,i,t,4] <- phi.ad[t]
    ps[3,i,t,5] <- 1-phi.ad[t]
    ps[4,i,t,1] <- 0
    ps[4,i,t,2] <- 0
    ps[4,i,t,3] <- 0
    ps[4,i,t,4] <- phi.ad[t]
    ps[4,i,t,5] <- 1-phi.ad[t]
    ps[5,i,t,1] <- 0
  }
}

```

```

ps[5,i,t,2] <- 0
ps[5,i,t,3] <- 0
ps[5,i,t,4] <- 0
ps[5,i,t,5] <- 1

# Define probabilities of O(t) given S(t)
po[1,i,t,1] <- 0
po[1,i,t,2] <- 0
po[1,i,t,3] <- 0
po[1,i,t,4] <- 1
po[2,i,t,1] <- 0
po[2,i,t,2] <- p.NB[t]
po[2,i,t,3] <- 0
po[2,i,t,4] <- 1-p.NB[t]
po[3,i,t,1] <- 0
po[3,i,t,2] <- p.NB[t]
po[3,i,t,3] <- 0
po[3,i,t,4] <- 1-p.NB[t]
po[4,i,t,1] <- 0
po[4,i,t,2] <- 0
po[4,i,t,3] <- p.B[t]
po[4,i,t,4] <- 1-p.B[t]
po[5,i,t,1] <- 0
po[5,i,t,2] <- 0
po[5,i,t,3] <- 0
po[5,i,t,4] <- 1
} #t
} #i

# Likelihood
for (i in 1:nind) {
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions) {
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], i, t-1,])
    } #t
  } #i
}
",fill=TRUE)
sink()

```

It is again possible to provide the known values of the latent states  $z$  as data and to estimate only the unknown values of  $z$ . However, this is trickier in this example because the labels of the observed states do not correspond with the labels of the latent states (see observation matrix above). For example, individuals that are observed and do not yet breed are labeled with "2", while they are in the latent state "2" or "3", depending on age. Furthermore, individuals that are observed breeding are labeled with "3", while they are in latent state "4". Some serious bookkeeping is

necessary to get the correct latent states for the observed capture-histories, and we avoid this here.

```

# Bundle data
bugs.data <- list(y = rCH, f = f, n.occasions = dim(rCH) [2], nind =
  dim(rCH) [1])

# Initial values (note: function ch.init is defined in section 7.3)
inits <- function() {list(mean.phi1 = runif(1, 0, 1), mean.phi2 =
  runif(1, 0, 1), mean.phiad = runif(1, 0, 1), mean.alpha1 = runif(1, 0,
  1), mean.alpha2 = runif(1, 0, 1), mean.pNB = runif(1, 0, 1), mean.pB =
  runif(1, 0, 1), z = ch.init(rCH, f))}

# Parameters monitored
parameters <- c("mean.phi1", "mean.phi2", "mean.phiad", "mean.alpha1",
  "mean.alpha2", "mean.pNB", "mean.pB")

# MCMC settings
ni <- 2000
nt <- 3
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 2 min)
agefirst <- bugs(bugs.data, inits, parameters, "agerecruitment.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

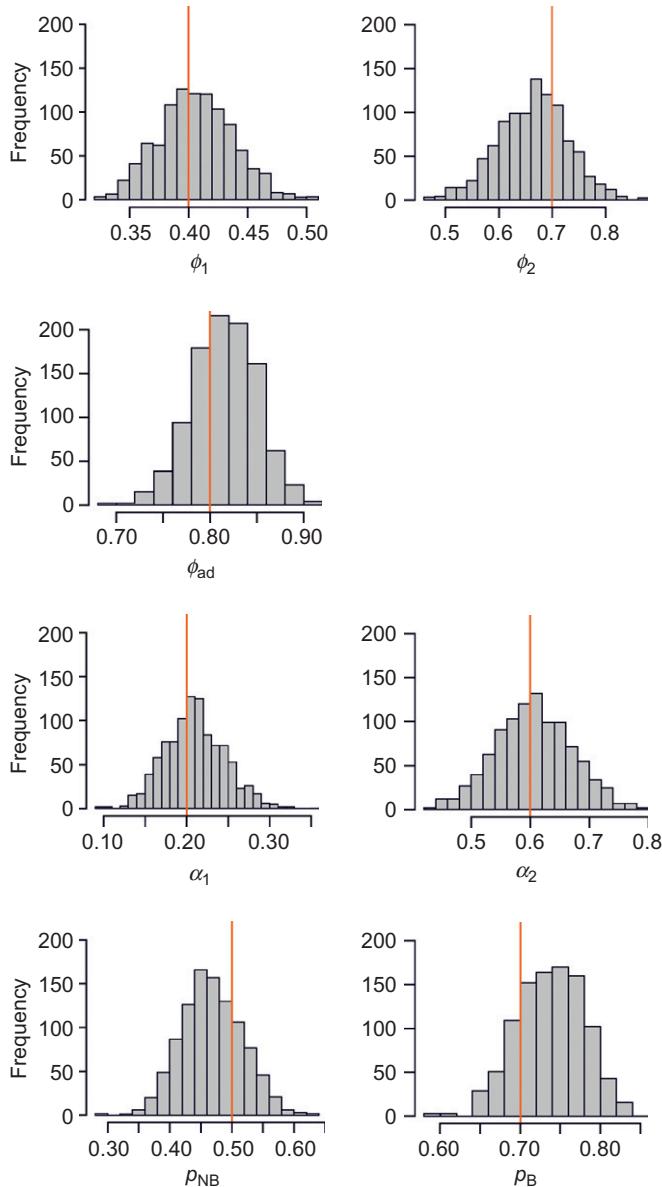
```

Convergence is achieved quickly, and all parameter estimates are reasonably precise (Fig. 9.6).

```

par(mfrow = c(3, 3), las = 1)
hist(agefirst$sims.list$mean.phi1, col = "gray", main = "",
  xlab = expression(phi[1]))
abline(v = phi.1, col = "red", lwd = 2)
hist(agefirst$sims.list$mean.phi2, col = "gray", main = "",
  xlab = expression(phi[2]), ylab = "")
abline(v = phi.2, col = "red", lwd = 2)
hist(agefirst$sims.list$mean.phiad, col = "gray", main = "",
  xlab = expression(phi[ad]), ylab = "")
abline(v = phi.ad, col = "red", lwd = 2)
hist(agefirst$sims.list$mean.alpha1, col = "gray", main = "",
  xlab = expression(alpha[1]))
abline(v = alpha.1, col = "red", lwd = 2)
hist(agefirst$sims.list$mean.alpha2, col = "gray", main = "",
  xlab = expression(alpha[2]), ylab = "")
abline(v = alpha.2, col = "red", lwd = 2)
plot(0, type = "n", axes = F, ylab = "", xlab = "")
hist(agefirst$sims.list$mean.pNB, col = "gray", main = "",
  xlab = expression(p[NB]))
abline(v = p.NB, col = "red", lwd = 2)
hist(agefirst$sims.list$mean.pB, col = "gray", main = "",
  xlab = expression(p[B]), ylab = "")
abline(v = p.B, col = "red", lwd = 2)

```



**FIGURE 9.6** Posterior distribution of survival, probability of breeding for the first time, and recapture. The vertical red lines show the values used for the simulations.

## 9.5 JOINT ANALYSIS OF CAPTURE–RECAPTURE AND MARK–RECOVERY DATA

### 9.5.1 Model Description

Both capture–recapture and mark–recovery data contain information about survival. Sometimes both data types are available in the same study, for example, in bird population studies (Frederiksen and Bregnballe, 2000; Altwein et al., 2007). It is then sensible to analyze them jointly to make use of all information about survival. The multinomial joint likelihood of this model was originally developed by Burnham (1993), Lebreton et al. (1995), and Barker (1997). More recently, Lebreton et al. (1999) showed how a joint analysis can be performed within the multistate modeling framework, and this is what we do here.

An important difference between capture–recapture and mark–recovery data is that they are often informative about two different kinds of survival probability. As we have seen in Chapter 7, capture–recapture data provide an estimate of apparent survival, which is the probability of surviving *and* remaining in the study area. By contrast, true survival can be estimated from mark–recovery data (Chapter 8). The difference between the two estimates of survival arises because sampling of marked individuals in capture–recapture studies is restricted to the study area (an exception might be studies using color marks), whereas dead recoveries can be obtained from anywhere. Apparent survival ( $\phi$ ) and true survival ( $s$ ) are linked through site fidelity ( $F$ ) via the relationship  $\phi = s^*F$ . A joint analysis allows one to estimate all three parameters.

The multistate model for the joint analysis of capture–recapture and mark–recovery data has four states: “alive in the study area”, “alive outside the study area”, “recently dead”, and “dead”. It may seem strange to have two dead states; however, this is necessary because only individuals recently dead can be recovered. An individual in the state “recently dead” at occasion  $t$  has died between occasions  $t - 1$  and  $t$ . From occasion  $t + 1$  onwards, the individual can no longer be recovered. This assumption applies to the analysis of mark–recovery data as well (Section 8.2). Therefore, “recently dead” individuals move to the state “dead” at the next occasion. One says that “dead” is an absorbing state; once individuals are in it, they cannot get out anymore. The transition matrix of the joint model looks like the following:

$$\begin{array}{ccccc}
 & \text{alive,} & \text{alive,} & \text{recently} & \\
 & \text{inside} & \text{outside} & \text{dead} & \text{dead} \\
 \text{alive, inside} & sF & s(1 - F) & 1 - s & 0 \\
 \text{alive, outside} & 0 & s & 1 - s & 0 \\
 \text{recently dead} & 0 & 0 & 0 & 1 \\
 \text{dead} & 0 & 0 & 0 & 1
 \end{array}$$

The parameters in the transition matrix are the true survival probability ( $s$ ) and fidelity probability ( $F$ ). Fidelity is defined as the probability to remain in the study area, given that an individual is alive. Thus, it is the complement of the probability to emigrate permanently from the study population ( $1 - F$ ). Permanent emigration denotes the permanent movement of individuals outside the study area. It is different from temporary emigration, which allows multiple exits and entries to the study population (see [Section 9.3](#)). The distinction between these two types of emigration is crucial.

The possible observations are “seen alive”, “recovered dead”, and “not seen or recovered”, and the observation matrix is

	seen alive	recovered dead	not seen or recovered
alive, inside	$p$	0	$1 - p$
alive, outside	0	0	1
recently dead	0	$r$	$1 - r$
dead	0	0	1

The parameters in the observation matrix are the recapture ( $p$ ) and the recovery probabilities ( $r$ ). The recapture probability is the probability to encounter an individual alive and in the study area, whereas the recovery probability is the probability to find and report an individual in the state “recently dead”. Both parameters are defined in the same way as the corresponding parameters of the CJS (Chapter 7) and the mark-recovery model (Chapter 8).

### 9.5.2 Generation of Simulated Data

We assume we want to estimate the adult survival of little ringed plovers ([Fig. 9.2](#)) that are marked with ordinary rings but also with colored wing tags, so that individuals can be encountered both live and dead. We imagine that adult survival ( $s$ ) is 0.8, and study site fidelity is high ( $F = 0.6$ ). The resighting probability is moderate ( $p = 0.5$ ), whereas the recovery probability is low ( $r = 0.1$ ). We simulate a study over 10 years, with 100 individuals marked each year.

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
s <- 0.8
F <- 0.6
r <- 0.1
p <- 0.5
n.occasions <- 10
n.states <- 4
n.obs <- 3
marked <- matrix(0, ncol=n.states, nrow=n.occasions)
marked[,1] <- rep(100, n.occasions) # Releases in study area
```

```

# Define matrices with survival, transition and recapture probabilities
# These are 4-dimensional matrices, with
  # Dimension 1: state of departure
  # Dimension 2: state of arrival
  # Dimension 3: individual
  # Dimension 4: time
# 1. State process matrix
totrel<- sum(marked) * (n.occasions-1)
PSI.STATE<- array(NA, dim=c(n.states, n.states, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.STATE[,,i,t] <- matrix(c(
      s*F, s*(1-F), 1-s, 0,
      0, s, 1-s, 0,
      0, 0, 0, 1,
      0, 0, 0, 1), nrow=n.states, byrow=TRUE)
  } #t
} #i

# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.OBS[,,i,t] <- matrix(c(
      p, 0, 1-p,
      0, 0, 1,
      0, r, 1-r,
      0, 0, 1), nrow=n.states, byrow=TRUE)
  } #t
} #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

# Compute date of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed!
# 1 = alive and in study are, 2 = recovered dead, 3 = not seen or recovered
rCH <- CH # Recoded CH
rCH[rCH==0] <- 3

```

### 9.5.3 Analysis of the Model

In principle, it should be possible to analyze the model in WinBUGS with code that directly translates the two matrices of the previous section, and thus in an analogous way as we have done for the other models in this chapter. However, due to a reason unknown to us, WinBUGS does not update the parameters properly, in particular the latent states  $z$ . It seems that the problem has to do with the recoveries; if they are excluded,

the problem goes away. By educated trial and error, we found out that a reparameterization of the model works fine, where the recovery probability is included in the state transition matrix:

$$\begin{bmatrix} s & s(1-F) & (1-s)r & (1-s)(1-r) \\ 0 & s & (1-s)r & (1-s)(1-r) \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The observation matrix then no longer contains the recovery probability:

$$\begin{bmatrix} p & 0 & 1-p \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This model is then fitted in WinBUGS:

```
# Specify model in BUGS language
sink("lifedeaf.bug")
cat("
model {

# -----
# Parameters:
# s: true survival probability
# F: fidelity probability
# r: recovery probability
# p: recapture/resighting probability
# -----
# States (S):
# 1 alive in study area
# 2 alive outside study area
# 3 recently dead and recovered
# 4 recently dead, but not recovered, or dead (absorbing)
# Observations (O):
# 1 seen alive
# 2 recovered dead
# 3 neither seen nor recovered
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)) {
  s[t] <- mean.s
  F[t] <- mean.f
  r[t] <- mean.r
  p[t] <- mean.p
}
mean.s ~ dunif(0, 1)      # Prior for mean survival
mean.f ~ dunif(0, 1)      # Prior for mean fidelity
```

```

mean.r ~ dunif(0, 1)      # Prior for mean recovery
mean.p ~ dunif(0, 1)      # Prior for mean recapture

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- s[t]*F[t]
    ps[1,i,t,2] <- s[t]*(1-F[t])
    ps[1,i,t,3] <- (1-s[t])*r[t]
    ps[1,i,t,4] <- (1-s[t])*(1-r[t])
    ps[2,i,t,1] <- 0
    ps[2,i,t,2] <- s[t]
    ps[2,i,t,3] <- (1-s[t])*r[t]
    ps[2,i,t,4] <- (1-s[t])*(1-r[t])
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 0
    ps[3,i,t,4] <- 1
    ps[4,i,t,1] <- 0
    ps[4,i,t,2] <- 0
    ps[4,i,t,3] <- 0
    ps[4,i,t,4] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- p[t]
    po[1,i,t,2] <- 0
    po[1,i,t,3] <- 1-p[t]
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- 0
    po[2,i,t,3] <- 1
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 1
    po[3,i,t,3] <- 0
    po[4,i,t,1] <- 0
    po[4,i,t,2] <- 0
    po[4,i,t,3] <- 1
  } #t
} #i

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], i, t-1,])
  } #t
} #i
}

",fill=TRUE)
sink()

```

```

# Bundle data
bugs.data <- list(y = rCH, f = f, n.occasions = dim(rCH)[2], nind = dim
  (rCH)[1])

# Initial values (note: function ch.init is defined in section 7.3)
inits <- function() {list(mean.s = runif(1, 0, 1), mean.f = runif(1, 0, 1),
  mean.p = runif(1, 0, 1), mean.r = runif(1, 0, 1), z = ch.init(CH, f))}

# Parameters monitored
parameters <- c("mean.s", "mean.f", "mean.r", "mean.p")

# MCMC settings
ni <- 40000
nt <- 10
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 80 min)
lifedeath <- bugs(bugs.data, inits, parameters, "lifedeath.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir)

```

The model does not converge easily, and long chains are necessary to obtain satisfactory results.

```

print(lifedeath, digit = 3)
      mean    sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
mean.s 0.843 0.051  0.746  0.808  0.841  0.882  0.935  1.004   590
mean.f 0.606 0.041  0.533  0.576  0.605  0.634  0.692  1.003   770
mean.r 0.145 0.045  0.089  0.114  0.133  0.162  0.262  1.004   730
mean.p 0.515 0.028  0.461  0.496  0.515  0.534  0.572  1.001   4600

```

This model can be extended in several ways, for instance, by including mortality causes (Schaub and Pradel, 2004; Schaub, 2009; Servanthy et al., 2010), movement between sites (Kendall et al., 2006; Duriez et al., 2009), or age structures, to name just a few. Obviously, a multistate model could be used when the goal is to analyze single-state capture–recapture data alone (the true states being “alive” and “dead”, and the observed states being “seen” and “not seen”), or if the goal is to analyze mark-recovery data alone (the true states being “alive”, “recently dead and recovered”, and “recently dead (not recovered) or dead”, and the observed states being “recovered” and “not recovered”).

## 9.6 ESTIMATION OF MOVEMENT AMONG THREE SITES

### 9.6.1 Model Description

In this example, we estimate movement among three sites. The list of states includes “alive at site A”, “alive at site B”, “alive at site C”, and “dead”, and the list of observations is “seen at A”, “seen at B”, “seen at C”, and “not seen”. The state transition matrix is

$$\begin{matrix}
 & \text{site A} & \text{site B} & \text{site C} & \text{dead} \\
 \text{site A} & \phi_A(1 - \psi_{AB} - \psi_{AC}) & \phi_A\psi_{AB} & \phi_A\psi_{AB} & 1 - \phi_A \\
 \text{site B} & \phi_B\psi_{BA} & \phi_B(1 - \psi_{BA} - \psi_{BC}) & \phi_B\psi_{BC} & 1 - \phi_B \\
 \text{site C} & \phi_C\psi_{CA} & \phi_C\psi_{CB} & \phi_C(1 - \psi_{CA} - \psi_{CB}) & 1 - \phi_C \\
 \text{dead} & 0 & 0 & 0 & 1
 \end{matrix}$$

and the observation matrix is

$$\begin{matrix}
 & \text{seen at A} & \text{seen at B} & \text{seen at C} & \text{not seen} \\
 \text{site A} & p_A & 0 & 0 & 1 - p_A \\
 \text{site B} & 0 & p_B & 0 & 1 - p_B \\
 \text{site C} & 0 & 0 & p_C & 1 - p_C \\
 \text{dead} & 0 & 0 & 0 & 1
 \end{matrix}$$

We here extend the multistate model for two sites (Section 9.2) because there is an additional challenge in the constraints that need to be put on the movement probabilities. For each site, we now have three movement probabilities. For instance, from site A, we have  $\psi_{AA}$ ,  $\psi_{AB}$ , and  $\psi_{AC}$ . Two sets of constraints must be imposed on their estimates: first, each of them must be in the interval  $[0, 1]$  and second, they must sum to 1. With only two transition probabilities as in all examples before, this is easy to achieve: we give a uniform or a beta prior to one of them and calculate the other as the complement to 1. With more than two states, this is no longer so easy, but there are two possible solutions: the use of a multinomial logit link function for the transition probabilities or the use of a Dirichlet distribution as a prior for the transition probabilities. Below, we illustrate both.

For a multinomial link function, we specify a normal prior distribution for  $n - 1$  transition parameters ( $\alpha$ ). These correspond to the transition probabilities on the logit scale, and they can be modeled using the GLM framework. This is exactly equivalent to the use of the logit function when the parameters on the logit scale are modeled with a GLM. The back-transformation for each of the  $n - 1$  parameters is calculated as

$$\beta_j = \frac{\exp(\alpha_j)}{1 + \sum_{i=1}^{n-1} \exp(\alpha_i)},$$

which ensures that each of them, as well as their sum, is  $< 1$ . The last parameter is calculated as  $\beta_n = 1 - \sum_{i=1}^{n-1} \beta_i$ . In this model, only  $n - 1$  parameters can be modeled directly with a GLM, and the last parameter is modified

indirectly, being a function of the covariates used in the GLM for the  $n - 1$  modeled parameters.

The second option is the use of a Dirichlet prior for the  $\beta$ s, which automatically ensures that they are in the interval  $[0, 1]$  and sum to 1. For the Dirichlet prior distribution, hyperparameters must be chosen. With three transition probabilities that must sum to one, a vector  $[1, 1, 1]$  induces a noninformative Dirichlet prior. This vector of hyperparameters is best given as data.

The Dirichlet prior for the  $\beta$ s can also be specified indirectly. This option works well in WinBUGS by specification of a set of gamma (1, 1) hyperprior random variables, say,  $\alpha_j \sim \text{gamma}(1, 1)$ , followed by the relation  $\beta_j = \alpha_j / \sum_{i=1}^n \alpha_i$  (Royle and Dorazio, 2008).

In the following example, we use the multinomial logit link function, whereas for the model in Section 9.7, we construct the Dirichlet prior distribution using gamma hyperpriors. The latter results in faster convergence, at least in our examples.

### 9.6.2 Generation of Simulated Data

We extend the previous little ringed plover example to three sites. We assume that habitat quality at site A is higher than at sites B and C, and thus annual survival at site A is higher ( $\phi_A = 0.85$ ) than at site B ( $\phi_B = 0.75$ ) and C ( $\phi_C = 0.65$ ). Furthermore, movement away from A is less likely ( $\psi_{AB} = 0.3$ ,  $\psi_{AC} = 0.2$ ) than movement to A ( $\psi_{BA} = 0.5$ ,  $\psi_{CA} = 0.6$ ). Movement rates between B and C are identical ( $\psi_{BC} = 0.1$ ,  $\psi_{CB} = 0.1$ ). Capture effort is greater at site A than at B or C resulting in the following recapture probabilities:  $p_A = 0.7$ ,  $p_B = 0.4$ , and  $p_C = 0.5$ . Captures at site A are labeled "1", at site B "2", and at site C "3".

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phia <- 0.85
phib <- 0.75
phic <- 0.65
psiAB <- 0.3
psiAC <- 0.2
psiba <- 0.5
psibc <- 0.1
psica <- 0.6
psicb <- 0.1
pA <- 0.7
pB <- 0.4
pC <- 0.5
```

```

n.occasions <- 6
n.states <- 4
n.obs <- 4
marked <- matrix(NA, ncol=n.states, nrow=n.occasions)
marked[,1] <- rep(50, n.occasions)
marked[,2] <- rep(50, n.occasions)
marked[,3] <- rep(50, n.occasions)
marked[,4] <- rep(0, n.occasions)

# Define matrices with survival, transition and recapture probabilities
# These are 4-dimensional matrices, with
  # Dimension 1: state of departure
  # Dimension 2: state of arrival
  # Dimension 3: individual
  # Dimension 4: time
# 1. State process matrix
totrel <- sum(marked)*(n.occasions-1)
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.STATE[,,i,t] <- matrix(c(
      phiA*(1-psiAB-psiAC), phiA*psiAB,           phiA*psiAC,           1-phiA,
      phiB*psiBA,                   phiB*(1-psiBA-psiBC), phiB*psiBC,           1-phiB,
      phiC*psiCA,                   phiC*psiCB,           phiC*(1-psiCA-psiCB), 1-phiC,
      0,                           0,                   0,                   1),
      nrow=n.states, byrow=TRUE)
    } #t
  } #i

# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.OBS[,,i,t] <- matrix(c(
      pA, 0, 0, 1-pA,
      0, pB, 0, 1-pB,
      0, 0, pC, 1-pC,
      0, 0, 0, 1), nrow=n.states, byrow=TRUE)
    } #t
  } #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

# Compute vector with occasions of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed in WinBUGS!
# 1 = seen alive in A, 2 = seen alive in B, 3, seen alive in C, 4 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 4

```

### 9.6.3 Analysis of the Model

We first write the model with the multinomial logit link.

```
# Specify model in BUGS language
sink("ms3-multinomlogit.bug")
cat("
model {

# -----
# Parameters:
# phiA: survival probability at site A
# phiB: survival probability at site B
# phiC: survival probability at site C
# psiAB: movement probability from site A to site B
# psiAC: movement probability from site A to site C
# psiBA: movement probability from site B to site A
# psiBC: movement probability from site B to site C
# psiCA: movement probability from site C to site A
# psiCB: movement probability from site C to site B
# pA: recapture probability at site A
# pB: recapture probability at site B
# pC: recapture probability at site C
# -----
# States (S):
# 1 alive at A
# 2 alive at B
# 3 alive at C
# 4 dead
# Observations (O):
# 1 seen at A
# 2 seen at B
# 3 seen at C
# 4 not seen
# -----
# Priors and constraints
# Survival and recapture: uniform
phiA ~ dunif(0, 1)
phiB ~ dunif(0, 1)
phiC ~ dunif(0, 1)
pA ~ dunif(0, 1)
pB ~ dunif(0, 1)
pC ~ dunif(0, 1)
# Transitions: multinomial logit
# Normal priors on logit of all but one transition prob.
for (i in 1:2) {
  lpsiA[i] ~ dnorm(0, 0.001)
  lpsiB[i] ~ dnorm(0, 0.001)
  lpsiC[i] ~ dnorm(0, 0.001)
}
# Constrain the transitions such that their sum is < 1
for (i in 1:2) {
  psiA[i] <- exp(lpsiA[i]) / (1 + exp(lpsiA[1]) + exp(lpsiA[2]))
}
```

```

psiB[i] <- exp(lpsiB[i]) / (1 + exp(lpsiB[1]) + exp(lpsiB[2]))
psiC[i] <- exp(lpsiC[i]) / (1 + exp(lpsiC[1]) + exp(lpsiC[2]))
}
# Calculate the last transition probability
psiA[3] <- 1-psiA[1]-psiA[2]
psiB[3] <- 1-psiB[1]-psiB[2]
psiC[3] <- 1-psiC[1]-psiC[2]

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- phiA * psiA[1]
    ps[1,i,t,2] <- phiA * psiA[2]
    ps[1,i,t,3] <- phiA * psiA[3]
    ps[1,i,t,4] <- 1-phiA
    ps[2,i,t,1] <- phiB * psiB[1]
    ps[2,i,t,2] <- phiB * psiB[2]
    ps[2,i,t,3] <- phiB * psiB[3]
    ps[2,i,t,4] <- 1-phiB
    ps[3,i,t,1] <- phiC * psiC[1]
    ps[3,i,t,2] <- phiC * psiC[2]
    ps[3,i,t,3] <- phiC * psiC[3]
    ps[3,i,t,4] <- 1-phiC
    ps[4,i,t,1] <- 0
    ps[4,i,t,2] <- 0
    ps[4,i,t,3] <- 0
    ps[4,i,t,4] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- pA
    po[1,i,t,2] <- 0
    po[1,i,t,3] <- 0
    po[1,i,t,4] <- 1-pA
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- pB
    po[2,i,t,3] <- 0
    po[2,i,t,4] <- 1-pB
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 0
    po[3,i,t,3] <- pC
    po[3,i,t,4] <- 1-pC
    po[4,i,t,1] <- 0
    po[4,i,t,2] <- 0
    po[4,i,t,3] <- 0
    po[4,i,t,4] <- 1
  } #t
} #i

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){

```

```

# State process: draw S(t) given S(t-1)
z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
# Observation process: draw O(t) given S(t)
y[i,t] ~ dcat(po[z[i,t], i, t-1,])
} #t
} #i
}
",fill=TRUE)
sink()

# Bundle data
bugs.data <- list(y=rCH, f=f, n.occasions = dim(rCH) [2], nind = dim
(rCH) [1], z=known.state.ms(rCH, 4))

# Initial values
inits <- function(){list(phiA=runif(1, 0, 1), phiB=runif(1, 0, 1),
phiC=runif(1, 0, 1), lpsiA=rnorm(2), lpsiB=rnorm(2), lpsiC=
rnorm(2), pA=runif(1, 0, 1) , pB=runif(1, 0, 1) , pC=runif(1, 0, 1),
z=ms.init.z(rCH, f))}

# Parameters monitored
parameters <- c("phiA", "phiB", "phiC", "psiA", "psiB", "psiC", "pA",
"pB", "pC")

# MCMC settings
ni <- 50000
nt <- 6
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT 56 min)
ms3 <- bugs(bugs.data, inits, parameters, "ms3-multinomlogit.bug",
n.chains=nc, n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE,
bugs.directory=bugs.dir, working.directory=getwd())

```

This model runs slowly and convergence is hard to achieve. To increase computation speed, we have provided the known values of the latent state  $z$  (see `bugs.data`). The parameter estimates look like the following:

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
phiA	0.833	0.029	0.780	0.813	0.832	0.851	0.893	1.001	8600
phiB	0.730	0.033	0.664	0.708	0.731	0.753	0.791	1.001	15000
phiC	0.676	0.036	0.607	0.651	0.676	0.699	0.747	1.001	15000
psiA[1]	0.546	0.080	0.395	0.491	0.546	0.602	0.702	1.001	15000
psiA[2]	0.310	0.070	0.173	0.262	0.310	0.357	0.449	1.002	1300
psiA[3]	0.144	0.040	0.091	0.115	0.135	0.165	0.240	1.007	320
psiB[1]	0.551	0.097	0.373	0.482	0.546	0.616	0.751	1.001	15000
psiB[2]	0.375	0.088	0.189	0.317	0.381	0.438	0.531	1.001	8400
psiB[3]	0.074	0.030	0.032	0.051	0.068	0.089	0.151	1.003	1000
psiC[1]	0.740	0.072	0.572	0.697	0.750	0.793	0.849	1.001	13000
psiC[2]	0.096	0.044	0.034	0.065	0.089	0.119	0.205	1.002	1300
psiC[3]	0.164	0.053	0.089	0.125	0.153	0.193	0.292	1.004	570
pA	0.671	0.085	0.535	0.610	0.662	0.720	0.870	1.001	15000
pB	0.456	0.132	0.280	0.366	0.427	0.514	0.817	1.002	1600
pC	0.732	0.170	0.398	0.601	0.749	0.877	0.988	1.005	450

## 9.7 REAL-DATA EXAMPLE: THE SHOWY LADY'S SLIPPER

Here, we want to estimate the survival probability of a plant species, the beautiful showy lady's slipper (Fig. 9.7). You may ask yourself why there is a need to apply multistate capture–recapture models to plant data; after all, don't plants just stand still and wait to be counted (Harper, 1977)? Couldn't we simply use logistic regression models for survival estimation? Well, this is only partly true. If a plant remains aboveground throughout its life, such as an oak, then this may be true. However, some plants are temporally unobservable because they are dormant and thus live underground for one or several years. Since individuals may also die when they are dormant, we have to use probabilistic models. Moreover, there is an interest in modeling the transition probabilities to and from the dormant state, as well as between different aboveground states, such as vegetative and reproductive. Knowing these transition probabilities is important for setting up matrix projection models for such species. Examples of the application of capture–recapture models (both single- and multistate models) for plants are Shefferson et al. (2001) and Kéry et al. (2005a).



**FIGURE 9.7** Showy lady's slipper (*Cypripedium reginae*), West Virginia, 2010 (Photograph by K. Gregg).

The multistate capture–recapture data analyzed here were collected by Kathy Gregg in Big Draft (West Virginia) from 1989 to 1999. Kéry and Gregg (2004) analyzed them in the frequentist framework. Three live states can be distinguished for the showy lady's slipper: dormant, vegetative, and flowering. We would like to estimate the probabilities of state transition and of survival of this species. For the state-transition matrix, we define four states: “vegetative”, “flowering”, “dormant”, and “dead”.

$$\begin{array}{cccc} & \text{vegetative} & \text{flowering} & \text{dormant} & \text{dead} \\ \text{vegetative} & s\psi_{VV} & s\psi_{VF} & s(1 - \psi_{VV} - \psi_{VF}) & 1 - s \\ \text{flowering} & s\psi_{FV} & s\psi_{FF} & s(1 - \psi_{FV} - \psi_{FF}) & 1 - s \\ \text{dormant} & s\psi_{DV} & s\psi_{DF} & s(1 - \psi_{DV} - \psi_{DF}) & 1 - s \\ \text{dead} & 0 & 0 & 0 & 1 \end{array}.$$

Permanent emigration is impossible, so we can estimate true survival; hence, the symbol  $s$  instead of  $\phi$ . For the observation matrix, we define the observations “seen vegetative”, “seen flowering”, and “not seen”. We assume that no plants were missed in the states “vegetative” and “flowering”, but they cannot be observed when they are either “dormant” or “dead”. Thus, the observation matrix is completely deterministic:

$$\begin{array}{ccc} & \text{seen veg.} & \text{seen flow.} & \text{not seen} \\ \text{vegetative} & 1 & 0 & 0 \\ \text{flowering} & 0 & 1 & 0 \\ \text{dormant} & 0 & 0 & 1 \\ \text{dead} & 0 & 0 & 1 \end{array}.$$

In the data, the vegetative state is coded as 1, the flowering state as 2, and failure to observe an individual as 0. We now read the data in an R workspace

```
CH <- as.matrix(read.table("orchids.txt", sep=" ", header=F))
n.occasions <- dim(CH)[2]

# Compute vector with occasion of first capture
f <- numeric()
for (i in 1:dim(CH)[1]) {f[i] <- min(which(CH[i, ]!=0))}

# Recode CH matrix: note, a 0 is not allowed by WinBUGS!
# 1 = seen vegetative, 2 = seen flowering, 3 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 3
```

Kéry and Gregg (2004) found that a model with year-specific survival and constant transition probabilities was most parsimonious, and we will use this parameter structure here. Since movement is possible among three live states, the transition probabilities must meet the constraints that each of them is between 0 and 1 and that they sum to 1. Here, we implement

the Dirichlet prior distribution using elemental gamma random variables as described in Section 9.6.1.

```
# Specify model in BUGS language
sink("ladyslipper.bug")
cat("
model {
# -----
# Parameters:
# s: survival probability
# psiV: transitions from vegetative
# psiF: transitions from flowering
# psiD: transitions from dormant
# -----
# States (S):
# 1 vegetative
# 2 flowering
# 3 dormant
# 4 dead
# Observations (O):
# 1 seen vegetative
# 2 seen flowering
# 3 not seen
# -----
# Priors and constraints
# Survival: uniform
for (t in 1:(n.occasions-1)){
  s[t] ~ dunif(0, 1)
}
# Transitions: gamma priors
for (i in 1:3){
  a[i] ~ dgamma(1, 1)
  psiD[i] <- a[i]/sum(a[])
  b[i] ~ dgamma(1, 1)
  psiV[i] <- b[i]/sum(b[])
  c[i] ~ dgamma(1, 1)
  psiF[i] <- c[i]/sum(c[])
}
# Define state-transition and observation matrices
for (i in 1:nind)
  # Define probabilities of state S(t+1) given S(t)
  for (t in 1:(n.occasions-1)){
    ps[1,i,t,1] <- s[t] * psiV[1]
    ps[1,i,t,2] <- s[t] * psiV[2]
    ps[1,i,t,3] <- s[t] * psiV[3]
    ps[1,i,t,4] <- 1-s[t]
    ps[2,i,t,1] <- s[t] * psiF[1]
    ps[2,i,t,2] <- s[t] * psiF[2]
    ps[2,i,t,3] <- s[t] * psiF[3]
    ps[2,i,t,4] <- 1-s[t]
```

```

ps[3,i,t,1] <- s[t] * psiD[1]
ps[3,i,t,2] <- s[t] * psiD[2]
ps[3,i,t,3] <- s[t] * psiD[3]
ps[3,i,t,4] <- 1-s[t]
ps[4,i,t,1] <- 0
ps[4,i,t,2] <- 0
ps[4,i,t,3] <- 0
ps[4,i,t,4] <- 1

# Define probabilities of O(t) given S(t)
po[1,i,t,1] <- 1
po[1,i,t,2] <- 0
po[1,i,t,3] <- 0
po[2,i,t,1] <- 0
po[2,i,t,2] <- 1
po[2,i,t,3] <- 0
po[3,i,t,1] <- 0
po[3,i,t,2] <- 0
po[3,i,t,3] <- 1
po[4,i,t,1] <- 0
po[4,i,t,2] <- 0
po[4,i,t,3] <- 1
} #t
} #i

# Likelihood
for (i in 1:nind){
  # Define latent state at first capture
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){

    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])

    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], i, t-1,])
  } #t
} #i
},fill=TRUE)
sink()

# Bundle data
bugs.data <- list(y=rCH, f=f, n.occasions=dim(rCH)[2],
  nind=dim(rCH)[1], z=known.state.ms(rCH, 3))

# Initial values
inits <- function(){list(s=runif((dim(rCH)[2]-1),0,1),
  z=ms.init.z(rCH, f))}

# Parameters monitored
parameters <- c("s", "psiV", "psiF", "psiD")

# MCMC settings
ni <- 5000
nt <- 3

```

```

nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
ls <- bugs(bugs.data, inits, parameters, "ladybug.bug", n.chains =
  nc, n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE,
  bugs.directory=bugs.dir, working.directory=getwd())
print(ls, digits = 3)
      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
s[1] 0.984 0.013 0.952 0.977 0.986 0.993 0.999 1.001 3000
[...]
s[10] 0.982 0.013 0.951 0.975 0.984 0.992 0.999 1.001 3000
psiV[1] 0.830 0.012 0.806 0.822 0.830 0.838 0.852 1.001 3000
psiV[2] 0.151 0.011 0.130 0.143 0.151 0.159 0.174 1.002 1400
psiV[3] 0.019 0.005 0.011 0.016 0.019 0.022 0.030 1.002 1800
psiF[1] 0.183 0.018 0.149 0.171 0.183 0.195 0.221 1.001 3000
psiF[2] 0.803 0.019 0.763 0.790 0.803 0.816 0.838 1.001 3000
psiF[3] 0.014 0.006 0.005 0.010 0.013 0.018 0.028 1.001 3000
psiD[1] 0.554 0.101 0.348 0.486 0.557 0.625 0.742 1.005 990
psiD[2] 0.172 0.069 0.059 0.122 0.166 0.212 0.327 1.001 3000
psiD[3] 0.274 0.099 0.106 0.200 0.265 0.337 0.484 1.002 2000

```

Note how the estimated transitions relate to those presented in the state-transition matrix above. The estimate  $\text{psiV}[1]$  is  $\psi_{VV}$ ,  $\text{psiV}[2]$  is  $\psi_{VF}$ ,  $\text{psiV}[3]$  is  $1 - \psi_{VV} - \psi_{VF}$ , and so on. The estimated parameter estimates match very well the results from the published frequentist analysis of the model for this data set (Kéry and Gregg, 2004).

## 9.8 SUMMARY AND OUTLOOK

In this chapter, we have introduced an important class of capture–recapture models: multistate models. Multistate models represent a very general framework and other models such as the CJS (Chapter 7), the mark-recovery (Chapter 8), and the JS model (Chapter 10) can be described as special cases (Lebreton et al., 1999, 2009). Multistate models are extremely flexible and can be used to address a large array of very diverse ecological questions. The flexibility stems from the fact that the definition of states can involve many different quantities, as seen in the examples, which go far beyond the classical geographic locations. In addition to the examples that we have provided in this chapter, we present some further models in Appendix 2 (including data simulation and BUGS model code). To analyze multistate models in the Bayesian framework, we use the state-space formulation.

Multistate capture–recapture models can be extended in exactly the same ways as we have shown for the single-state models. Thus, we may model several groups, time-dependent parameters, and individual or temporal covariates, and group effects can be assumed fixed or

random. The data and most parts of the BUGS model code remain the same, but there are modifications in the “Priors and constraints” part of the model, in exactly the same way as was shown in Chapters 6 and 7. The exercises and their solutions provide some examples of such extensions. Multistate models have also been combined with other model classes, such as site-occupancy models (Section 13.6).

Multistate capture–recapture models sometimes suffer from identifiability problems, both intrinsic and extrinsic. Hence, it is important to check for identifiability, in particular when adopting less well-known multistate model variants. To check identifiability, the method introduced in Section 7.9 can be used. Moreover, we recommend assessing the goodness of fit using  $\chi^2$ -decompositions developed by Pradel et al. (2003).

Recently, an extremely general extension of the general multistate model to include state uncertainty has been developed (Pradel, 2005). E-SURGE (Choquet et al. 2009b) is a flexible software to fit such models using maximum likelihood. These so-called multievent models belong to the class of hidden Markov models and can also be fitted using WinBUGS. An important difference is that in the conventional multistate models of this chapter, there is no error in the state assignment, whereas in multievent models, state assignment errors can occur. In other words, multistate models allow only false-negative errors, while multievent models allow both false-negative and false-positive errors. The state-space formulation used here to fit multistate models can be adapted to include state assignment errors; see also Section 13.6. The observation matrix must then be written in such a way that false state assignments are possible. A further difference is that there is also uncertainty about the state at first observation in the multievent model. We can model this uncertainty by estimating a probability of state membership at first encounter based on the observed states. This means that we have several additional parameters in the model that require priors.

## 9.9 EXERCISES

1. Simulate multistate capture–recapture data for two sexes (m, f) in two populations (A, B) that are connected by dispersal. Assume that movement rates between populations are the same for both sexes, but that site-specific survival and recapture differ among populations. The simulation parameters are  $\phi_{A,m} = 0.5$ ,  $\phi_{B,m} = 0.6$ ,  $\phi_{A,f} = 0.7$ ,  $\phi_{B,f} = 0.6$ ,  $\psi_{AB} = 0.2$ ,  $\psi_{BA} = 0.5$ ,  $p_{A,m} = 0.3$ ,  $p_{B,m} = 0.7$ ,  $p_{A,f} = 0.4$ ,  $p_{B,f} = 0.8$ , occasions = 6, and 20 males and females are released at each population in each year. Simulate the data and analyze them.
2. Simulate multistate capture–recapture data from two populations observed over 8 occasions that exchange individuals with the following

parameter values:  $\phi_A = [0.5, 0.6, 0.3, 0.7, 0.5, 0.65, 0.55]$ ,  $\phi_B = 0.6$ ,  $\psi_{AB} = 0.2$ ,  $\psi_{BA} = 0.5$ ,  $p_A = 0.3$ ,  $p_B = 0.7$ , and 20 individuals are released at each population in each year. Thus, we assume that the annual survival probabilities vary among years at location A, but not at location B. Simulate data and analyze them, a) assuming fixed year effects and b) assuming random year effects.

3. In a population of salamanders, there is nonrandom temporary emigration (with respect to one breeding site). In addition, there is strong individual heterogeneity in capture probability. Assume a 10-years study and the following parameter values: survival = 0.7,  $\psi_{IO} = 0.4$ ,  $\psi_{OI} = 0.8$ , mean recapture = 0.5, and the variance among individuals of the logit of recapture  $\sigma_i^2 = 0.4$ . Further assume that 100 salamanders are newly marked each year. Simulate data with these characteristics and analyze them.

## 10

# Estimation of Survival, Recruitment, and Population Size from Capture–Recapture Data Using the Jolly–Seber Model

## OUTLINE

10.1 Introduction	316
10.2 The JS Model as a State-Space Model	317
10.3 Fitting the JS Model with Data Augmentation	319
10.3.1 <i>The JS Model as a Restricted Dynamic Occupancy Model</i>	320
10.3.2 <i>The JS Model as a Multistate Model</i>	322
10.3.3 <i>The Superpopulation Parameterization</i>	325
10.4 Models with Constant Survival and Time-Dependent Entry	328
10.4.1 <i>Analysis of the JS Model as a Restricted Occupancy Model</i>	331
10.4.2 <i>Analysis of the JS Model as a Multistate Model</i>	332
10.4.3 <i>Analysis of the JS Model Under the Superpopulation Parameterization</i>	333
10.4.4 <i>Comparison of Estimates</i>	333
10.5 Models with Individual Capture Heterogeneity	335
10.6 Connections between Parameters, Further Quantities and Some Remarks on Identifiability	339
10.7 Analysis of a Real Data Set: Survival, Recruitment and Population Size of Leisler’s Bats	341
10.8 Summary and Outlook	345
10.9 Exercises	346

## 10.1 INTRODUCTION

Capture–recapture data contain information not only about the disappearance of marked individuals from a study population (i.e., mortality and permanent emigration) but also about their arrival into the population (i.e., recruitment by locally born individuals and immigration). Estimation of these recruitment-related parameters is possible when the *complete* capture-histories of the marked individuals are analyzed, not just the part following first capture, as for the CJS model (Chapter 7). The leading capture-history zeros (those before initial capture) along with the first capture contain the information about the arrival process. A leading zero is observed either because an individual has not yet arrived (recruited or immigrated) in the population or because it was already in the population but has not been captured. The part of the capture-histories after the initial capture contains information about survival, which is modeled by the CJS model (Chapter 7). All open-population capture–recapture types of models described in the Chapters 7–9 condition on first capture. This means that only the part of a capture-history *after* first capture is modeled; the information about entry of individuals is discarded. In this chapter, we introduce the Jolly–Seber (JS) model (Jolly, 1965; Seber, 1965), which allows estimation of gains to and losses from the population by *not* conditioning on first capture. Additionally, population size and the number of recruits per occasion can be estimated. The JS model is an extension of the closed-population models in Chapter 6, which do not condition on first capture either but assume demographic closure (i.e., the absence of mortality, recruitment, or dispersal), and of the CJS model.

Since the JS model uses the complete capture-history, additional assumptions are necessary. All assumptions listed for the CJS model (Section 7.1) also apply to the JS model. In addition, the JS model requires that *all* individuals (marked *and* unmarked) in the population have the same capture probability. In other words, newly captured individuals must be a random sample of all unmarked individuals in a population (Williams et al., 2002). This assumption is likely to be met when the same sampling protocol is applied for capture and recapture, but not when protocols differ. In many studies where color marks are applied, initial capture is a physical capture, whereas recaptures are just sightings from a distance. In this case, the equal-capturability assumption is likely to be violated. Moreover, the assumption of homogeneous capture is violated in the presence of a permanent trap effect. Violation of the equal-catchability assumption biases estimates of population size and recruitment, but not of survival (Nichols et al., 1984). Therefore, permanent trap effects do not violate the CJS assumptions. Williams et al. (2002) provide an in-depth overview of JS model assumptions and consequences of their failure for the parameter estimates.

There are a number of different parameterizations of the JS model, which differ basically in the way how recruitment is modeled. They include the original parameterizations of Jolly (1965) and Seber (1965), the superpopulation approach (Crosbie and Manly, 1985; Schwarz and Arnason, 1996), the reverse-time formulation (Pradel, 1996), the parameterization of Link and Barker (2005), and finally the formulation as a restricted dynamic occupancy model (Royle and Dorazio, 2008). All JS model parameterizations are related and should give the same estimates of population size and recruitment parameters. Here, we illustrate three JS model variants: the restricted occupancy formulation, the description as a multistate model, and the superpopulation formulation (Royle and Dorazio, 2008). We make extensive use of parameter-expanded data augmentation (introduced in Chapter 6) because it makes the modeling much easier (Royle et al., 2007a; Royle and Dorazio, 2011).

Numerous JS model variants have been described in the frequentist framework (e.g., Pollock et al., 1990; Pradel, 1996; Schwarz and Arnason, 1996; Williams et al., 2002) and implemented in software such as MARK (White and Burnham, 1999) or POPAN (Arnason and Schwarz, 1999). In contrast, Bayesian applications are still relatively rare (but see Link and Barker, 2005; Dupuis and Schwarz, 2007; Schofield and Barker, 2008; Royle and Dorazio, 2008; Gardner et al., 2010; Link and Barker, 2010; Royle and Dorazio, 2011).

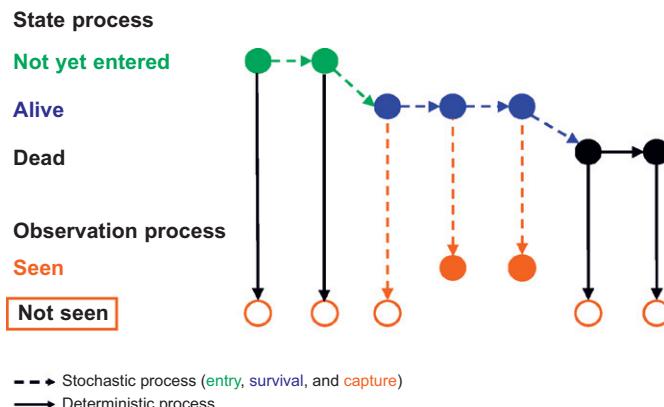
## **10.2 THE JS MODEL AS A STATE-SPACE MODEL**

We formulate the JS model in the hierarchical state-space framework. As mentioned repeatedly, the observed capture–recapture data are described as the result of a state process (the ecological process) and the observation process, which depends on the result of the state process. We denote the number of individuals ever alive during the study  $N_s$  and call it the superpopulation size (Schwarz and Arnason, 1996). We further assume that a fraction of  $N_s$  is already alive and in the study area at the first capture occasion, and that all remaining individuals have entered the population by the end of the study. The probability that a member of  $N_s$  enters the population at occasion  $t$  is  $b_t$  ( $t = 1, \dots, T$ ) and is called the entry probability (Schwarz and Arnason, 1996). It is the probability that an individual is new in the population, that is, it has entered the population since the preceding occasion. Entry could result either from in situ recruitment (locally born individuals) or from immigration. Sometimes the entry probability is called recruitment probability, but this is inaccurate. The number of individuals entering the population at  $t$  is  $B_t = N_s b_t$ . The fraction of individuals already present at the first occasion is  $b_1$ ; this “entry” probability has no clear ecological meaning because it is a complex function of all entries before the first occasion. All entry probabilities must sum to 1 to

ensure that all  $N_s$  individuals enter the population sometime during the study. The number of individuals entering at each occasion can be modeled with a multinomial distribution as  $\mathbf{B} \sim \text{multinomial}(N_s, \mathbf{b})$  (note that  $\mathbf{B}$  and  $\mathbf{b}$  are vectors).

As in the CJS model, we denote the latent state of individual  $i$  at occasion  $t$  as  $z_{i,t} = 1$ , if it is alive and present in the population, and as  $z_{i,t} = 0$ , if it is either dead or has not yet entered the population. Thus, if individual  $i$  enters the population at  $t$ , its latent state changes from  $z_{i,t-1} = 0$  to  $z_{i,t} = 1$  (Fig. 10.1). On entry, the survival process starts, which is a simple coin flip and identical to that in the CJS model (Chapter 7). Technically, the latent state  $z_{i,t+1}$  at  $t + 1$  is determined by a Bernoulli trial with success probability  $\phi_{i,t}$  ( $t = 1, \dots, T - 1$ ). The two processes defined so far, the entry and the survival process, represent the latent state processes. The observation process is defined for individuals that are alive ( $z = 1$ ). As usual, we assume that the detection of individual  $i$  at occasion  $t$  is determined by another coin flip with success probability  $p_{i,t}$  ( $t = 1, \dots, T$ ), that is, by another Bernoulli trial.

The resulting capture–recapture data consist of the capture-histories of  $n$  individuals. When capture probability is  $< 1$ , typically not all individuals in a population are captured; hence,  $n < N_s$ . If  $N_s$  were known, the capture–recapture data would contain in addition  $N_s - n$  all-zero capture-histories, and the model specification would be relatively easy. We could just use a multinomial distribution to estimate entry probabilities. However,  $N_s$  is unknown and so the multinomial index is also unknown and must be estimated. Moreover, parameters such as entry and capture probabilities refer to the complete population ( $N_s$ ), not just

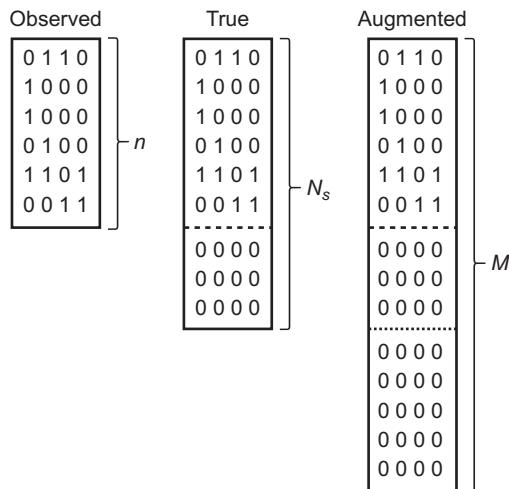


**FIGURE 10.1** Example of the state and observation process of a marked individual over time in the JS model. The sequence of true states for this individual under the restricted occupancy and superpopulation formulation is  $z = [0, 0, 1, 1, 1, 0, 0]$ , and the observed capture-history is  $y = [0, 0, 0, 1, 1, 0, 0]$ . In the multistate formulation,  $z = [1, 1, 2, 2, 2, 3, 3]$  and  $y = [2, 2, 2, 1, 1, 2, 2]$ .

to the  $n$  individuals ever captured. To deal with these challenges, we use parameter-expanded data augmentation (PX-DA; Tanner and Wong, 1987; Liu and Wu, 1999; Royle et al., 2007a; Royle and Dorazio, 2011), as we did in Chapter 6. The key idea is to fix the dimension of the parameter space in the analysis by augmenting the observed data with a large number of all-zero capture-histories, resulting in a larger data set of fixed dimension  $M$ , and to analyze the augmented data set using a reparameterized (zero-inflated) version of the model that would be applied if  $N_s$  were known. For a useful recent review of PX-DA, see Royle and Dorazio (2011).

## 10.3 FITTING THE JS MODEL WITH DATA AUGMENTATION

After augmentation, the capture–recapture data set contains  $M$  individuals, of which  $N_s$  are genuine and  $M-N_s$  are pseudo-individuals (Fig. 10.2). We do not know the proportions of genuine and pseudo-individuals, but we can estimate them. There are different ways to parameterize a JS model, and we present three here. First, the entry to the population is described as a removal process from  $M$  so that at the end of the study,  $N_s$  ( $N_s \leq M$ ) individuals have entered. This model can be developed either as a restricted version of a dynamic occupancy model



**FIGURE 10.2** Observed, true, and augmented capture-history matrix. The observed number of capture-histories is  $n$ , true population size  $N_s$ , and the size of the augmented data set  $M$ ;  $n$  and  $M$  are known, and  $N_s$  can be estimated.

(Section 13.5) or as a multistate model (Chapter 9). As a third approach, we use a zero-inflated version of the superpopulation formulation.

### 10.3.1 The JS Model as a Restricted Dynamic Occupancy Model

Royle and Dorazio (2008) showed that the JS model can be formulated as a restricted dynamic occupancy model. The entry process is defined slightly differently from what we described earlier. We imagine that individuals can be in one of three possible states: “not yet entered”, “alive”, and “dead” (Fig. 10.1). The state transitions are governed by two ecological processes, entry and survival, which we estimate. We denote as  $\gamma_t$  ( $t = 1, \dots, T$ ), the probability that an available individual in  $M$  enters the population at occasion  $t$ . This corresponds to the transition probability from state “not yet entered” to the state “alive”. Importantly,  $\gamma$  refers to available individuals, that is, to those in  $M$  that have not yet entered. The entry process is thus a removal process; over time, fewer and fewer individuals will be in the state “not yet entered” and thus available to entering the population. As a result,  $\gamma$  will increase over time on average, even with constant per-capita recruitment. It is a pure “nuisance parameter”, which is needed to describe the system, but without an ecological meaning. We refer to  $\gamma$  as a removal entry probability. The expected number of individuals present at the first occasion is  $E(B_1) = M\gamma_1$ . The expected number of individuals entering at the second occasion is the product of the number of individuals still available to enter and  $\gamma_2$ , thus  $E(B_2) = M(1 - \gamma_1)\gamma_2$ . More generally, the expected number of individuals entering the population at  $t$  is  $E(B_t) = M \prod_{i=1}^{t-1} (1 - \gamma_i)\gamma_t$ , and the total number of individuals that ever enter is  $N_s = \sum B$ .

The state of individual  $i$  at the first occasion is

$$z_{i,1} \sim \text{Bernoulli}(\gamma_1).$$

Subsequent states are determined either by survival, for an individual already entered ( $z_{i,t} = 1$ ), or by entry for one that has not ( $z_{i,t} = 0$ ). Thus,

$$z_{i,t+1} | z_{i,t}, \dots, z_{i,1} \sim \text{Bernoulli}\left(z_{i,t}\phi_{i,t} + \gamma_{t+1} \prod_{k=1}^t (1 - z_{i,k})\right).$$

Survival probability between occasion  $t$  and  $t + 1$  for individual  $i$  is denoted as  $\phi_{i,t}$ . In contrast,  $\gamma_t$  is only indexed by time because an index for individual would not be meaningful. The two equations above describe the state process of the JS model. The observation process is the same as in the CJS model (Section 7.2). It conditions on the state process and is

$$y_{i,t} | z_{i,t} \sim \text{Bernoulli}(z_{i,t}p_{i,t}).$$

This model is very similar to the dynamic occupancy model (Section 13.5). In the JS model, we just need to add a constraint that recruitment is not possible after death, whereas in occupancy models, recolonization after local extinction is possible.

Several quantities of interest can be derived from the latent state variable  $z$ . Population size at  $t$  is  $N_t = \sum_{i=1}^M z_{i,t}$ , the number of “fresh recruits” (newly entered individuals) at  $t$  is  $B_t = \sum_{i=1}^M (1 - z_{i,t-1})z_{i,t}$ , and superpopulation size is  $N_s = \sum \mathbf{B}$ .

In a Bayesian analysis of the model, the following parameters require priors: survival ( $\phi$ ), capture ( $p$ ) and removal entry probabilities ( $\gamma$ ). To express ignorance, we might specify a uniform prior  $U(0, 1)$  on all of them. The superpopulation size is a derived parameter; hence, a prior for  $N_s$  is defined implicitly, but it is not clear how it would have to look like. In any case, it is not a discrete uniform prior such as  $U(0, M)$ , which is what we might like to use (Royle and Dorazio, 2008), and what we use in the closed-population case (Chapter 6).

The BUGS code for this model is not difficult conceptually, but relatively long, because several quantities of interest are calculated (all after “Calculate derived population parameters”). This could be removed from the code and calculated outside BUGS; it just requires monitoring of the latent variable  $z$ . We present the code of a model with constant survival and recapture probabilities and time-dependent recruitment (entry).

```
# Specify model in BUGS language
sink("js-rest.occ.bug")
cat("
model {

# Priors and constraints
for (i in 1:M) {
  for (t in 1:(n.occasions-1)) {
    phi[i,t] <- mean.phi
  } #t
  for (t in 1:n.occasions) {
    p[i,t] <- mean.p
  } #t
} #i
mean.phi ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

for (t in 1:n.occasions) {
  gamma[t] ~ dunif(0, 1)
} #t

# Likelihood
for (i in 1:M) {
  # First occasion
  # State process
```

```

z[i,1] ~ dbern(gamma[1])
mu1[i] <- z[i,1] * p[i,1]
# Observation process
y[i,1] ~ dbern(mu1[i])
# Subsequent occasions
for (t in 2:n.occasions){
  # State process
  q[i,t-1] <- 1-z[i,t-1]           # Availability for recruitment
  mu2[i,t] <- phi[i,t-1] * z[i,t-1] + gamma[t] * prod(q[i,1:(t-1)])
  z[i,t] ~ dbern(mu2[i,t])
  # Observation process
  mu3[i,t] <- z[i,t] * p[i,t]
  y[i,t] ~ dbern(mu3[i,t])
} #t
} #i

# Calculate derived population parameters
for (t in 1:n.occasions){
  qgamma[t] <- 1-gamma[t]
}
cprob[1] <- gamma[1]
for (t in 2:n.occasions){
  cprob[t] <- gamma[t] * prod(qgamma[1:(t-1)])
} #t
psi <- sum(cprob[])                  # Inclusion probability
for (t in 1:n.occasions){
  b[t] <- cprob[t] / psi            # Entry probability
} #t
for (i in 1:M){
  recruit[i,1] <- z[i,1]
  for (t in 2:n.occasions){
    recruit[i,t] <- (1-z[i,t-1]) * z[i,t]
  } #t
} #i
for (t in 1:n.occasions){
  N[t] <- sum(z[1:M,t])            # Actual population size
  B[t] <- sum(recruit[1:M,t])      # Number of entries
} #t
for (i in 1:M){
  Nind[i] <- sum(z[i,1:n.occasions])
  Nalive[i] <- 1-equals(Nind[i], 0)
} #i
Nsuper <- sum(Nalive[])              # Superpopulation size
}
",fill=TRUE)
sink()

```

### 10.3.2 The JS Model as a Multistate Model

It is obvious from Fig. 10.1 that the JS model can also be viewed as a multistate model (Royle and Dorazio, 2011). This formulation has the

advantage that it can be extended flexibly to include age classes, multiple sites, dead recoveries, or others, as we showed in Chapter 9. To fit the JS as a multistate model in BUGS, we again first have to define lists of true and observed states and to describe the transitions between them with appropriate parameters. For the JS model, the true states are “not yet entered”, “alive”, and “dead” (Fig. 10.1), and the state transition matrix therefore is

$$\begin{array}{ccccc} & & \text{not yet entered} & \text{alive} & \text{dead} \\ \text{not yet entered} & & \begin{bmatrix} 1 - \gamma & \gamma & 0 \\ 0 & \phi & 1 - \phi \\ 0 & 0 & 1 \end{bmatrix} \\ \text{alive} & & & & \\ \text{dead} & & & & \end{array}$$

The parameters are the same as in the restricted occupancy formulation; thus,  $\gamma$  is the removal entry probability and  $\phi$  is survival. The observation process maps the three true states on the two observed states “seen” and “not seen”:

$$\begin{array}{ccc} & \text{seen} & \text{not seen} \\ \text{not yet entered} & \begin{bmatrix} 0 & 1 \\ p & 1 - p \\ 0 & 1 \end{bmatrix} \\ \text{alive} & & \\ \text{dead} & & \end{array}$$

The implementation as a state-space model works similarly as for other multistate models (Chapter 9). However, since the traditional multistate models condition on initial capture, there is no way to estimate  $\gamma_1$  at the first occasion. This can be overcome easily by adding a dummy occasion that contains only “0” before the first real occasion in the data. In the model specification, we then need to ensure that all individuals in the augmented data set are in state “not yet entered” at this first dummy occasion with probability 1. In this way, we solve two problems. First, the proportion of individuals present already at the first real occasion is estimated by the first transition, and second, the analyzed capture-histories condition on the first dummy occasion, which means that the model becomes unconditional for all real occasions.

The BUGS code to implement this model is given below. As discussed earlier, we assume constant survival and capture and time-dependent entry probability. The same quantities as before can be derived, we just have to remember that the latent state variable  $z$  now takes values 1 (“not yet entered”), 2 (“alive”), and 3 (“dead”), and thus these quantities must be calculated slightly differently. Priors are specified in the same way as for the restricted occupancy formulation.

```
# Specify model in BUGS language
sink("js-ms.bug")
cat("
model {

# -----
# Parameters:
# phi: survival probability
# gamma: removal entry probability
# p: capture probability
# -----
# States (S):
# 1 not yet entered
# 2 alive
# 3 dead
# Observations (O):
# 1 seen
# 2 not seen
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi[t] <- mean.phi
  gamma[t] ~ dunif(0, 1) # Prior for entry probabilities
  p[t] <- mean.p
}

mean.phi ~ dunif(0, 1)      # Prior for mean survival
mean.p ~ dunif(0, 1)        # Prior for mean capture

# Define state-transition and observation matrices
for (i in 1:M){
  # Define probabilities of state S(t+1) given S(t)
  for (t in 1:(n.occasions-1)){
    ps[1,i,t,1] <- 1-gamma[t]
    ps[1,i,t,2] <- gamma[t]
    ps[1,i,t,3] <- 0
    ps[2,i,t,1] <- 0
    ps[2,i,t,2] <- phi[t]
    ps[2,i,t,3] <- 1-phi[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- 0
    po[1,i,t,2] <- 1
    po[2,i,t,1] <- p[t]
    po[2,i,t,2] <- 1-p[t]
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 1
  } #t
} #i

# Likelihood
for (i in 1:M) {
```

```

# Define latent state at first occasion
z[i,1] <- 1      # Make sure that all M individuals are in state 1 at t=1
for (t in 2:n.occasions) {
    # State process: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation process: draw O(t) given S(t)
    y[i,t] ~ dcat(po[z[i,t], i, t-1,])
} #t
} #i

# Calculate derived population parameters
for (t in 1:(n.occasions-1)) {
    qgamma[t] <- 1-gamma[t]
}
cprob[1] <- gamma[1]
for (t in 2:(n.occasions-1)) {
    cprob[t] <- gamma[t] * prod(qgamma[1:(t-1)])
} #t
psi <- sum(cprob[])          # Inclusion probability
for (t in 1:(n.occasions-1)) {
    b[t] <- cprob[t] / psi      # Entry probability
} #t

for (i in 1:M) {
    for (t in 2:n.occasions) {
        al[i,t-1] <- equals(z[i,t], 2)
    } #t
    for (t in 1:(n.occasions-1)) {
        d[i,t] <- equals(z[i,t]-al[i,t],0)
    } #
    alive[i] <- sum(al[i,])
} #i

for (t in 1:(n.occasions-1)) {
    N[t] <- sum(al[,t])          # Actual population size
    B[t] <- sum(d[,t])           # Number of entries
} #t
for (i in 1:M) {
    w[i] <- 1-equals(alive[i],0)
} #i
Nsuper <- sum(w[])            # Superpopulation size
}
",fill=TRUE)
sink()

```

### 10.3.3 The Superpopulation Parameterization

An important parameterization of the JS model was developed by Crosbie and Manly (1985), extended by Schwarz and Arnason (1996), and implemented as a hierarchical model by Royle and Dorazio (2008) and Link and Barker (2010). This parameterization uses entry probabilities  $b$  and an inclusion parameter  $\psi$ . To keep the sequential specification of the

state process model, we reexpress the entry probabilities ( $b$ ) as conditional entry probabilities ( $\eta$ ), and thus

$$\eta_1 = b_1, \quad \eta_2 = \frac{b_2}{1 - b_1}, \quad \dots, \quad \eta_t = \frac{b_t}{1 - \sum_{i=1}^{t-1} b_i}$$

The conditional entry probabilities ( $\eta$ ) are not the same as the removal entry probabilities ( $\gamma$ ) in Sections 10.3.1 and 10.3.2. Nevertheless, the state process is identical to that in the restricted occupancy parameterization, except that it contains  $\eta$  instead of  $\gamma$ . For the observation process, we suppose that each individual of  $M$  has an associated latent variable  $w_i \sim \text{Bernoulli}(\psi)$ . Individuals with  $w_i = 1$  are exposed to sampling if alive, whereas individuals with  $w_i = 0$  are not exposed to sampling. Thus, the observation model is

$$y_{i,t} | z_{i,t} \sim \text{Bernoulli}(w_i z_{i,t} p_{i,t}).$$

This observation model formally admits the zero-inflation of the augmented data set.

Here, we can derive some population estimates of interest as well. The vector of latent state variables  $z$  is inflated under the superpopulation formulation because it has length  $M$ , rather than  $N_s$ . We calculate  $u_{i,t} = z_{i,t} w_i$  to account for this. By using  $u$  instead of  $z$ , we can use the same formulas as for the restricted occupancy formulation to calculate the derived population estimates.

We need to specify priors for the survival and capture probabilities as before, but not for the conditional entry probabilities ( $\eta$ ). Instead, we specify priors for the entry probabilities ( $b$ ), and a convenient one is the Dirichlet prior:  $b_t \sim \text{Dirichlet}(\alpha)$ . To express ignorance and allocate the entries of all individuals uniformly over the  $T$  occasions, we set  $\alpha_t = 1$  for all  $t$ . In addition, we give a  $U(0, 1)$  prior for the inclusion probability  $\psi$ . This induces a discrete  $U(0, M)$  prior for the superpopulation size  $N_s$  (Royle and Dorazio, 2008), as we did for the closed-population models (Chapter 6).

The BUGS code for the superpopulation parameterization again has constant survival and capture probabilities and time-dependent entry probabilities.

```
# Specify model in BUGS language
sink("js-super.bug")
cat("
model {

# Priors and constraints
for (i in 1:M) {
  for (t in 1:(n.occasions-1)) {
    phi[i,t] <- mean.phi
  } #t
}
```

```

for (t in 1:n.occasions) {
  p[i,t] <- mean.p
} #t
} #i

mean.phi ~ dunif(0, 1)           # Prior for mean survival
mean.p ~ dunif(0, 1)             # Prior for mean capture
psi ~ dunif(0, 1)                # Prior for inclusion probability

# Dirichlet prior for entry probabilities
for (t in 1:n.occasions) {
  beta[t] ~ dgamma(1, 1)
  b[t] <- beta[t] / sum(beta[1:n.occasions])
}

# Convert entry probs to conditional entry probs
nu[1] <- b[1]
for (t in 2:n.occasions) {
  nu[t] <- b[t] / (1-sum(b[1:(t-1)]))
} #t

# Likelihood
for (i in 1:M) {
  # First occasion
  # State process
  w[i] ~ dbern(psi)           # Draw latent inclusion
  z[i,1] ~ dbern(nu[1])
  # Observation process
  mu1[i] <- z[i,1] * p[i,1] * w[i]
  y[i,1] ~ dbern(mu1[i])

  # Subsequent occasions
  for (t in 2:n.occasions) {
    # State process
    q[i,t-1] <- 1-z[i,t-1]
    mu2[i,t] <- phi[i,t-1] * z[i,t-1] + nu[t] * prod(q[i,1:(t-1)])
    z[i,t] ~ dbern(mu2[i,t])
    # Observation process
    mu3[i,t] <- z[i,t] * p[i,t] * w[i]
    y[i,t] ~ dbern(mu3[i,t])
  } #t
} #i

# Calculate derived population parameters
for (i in 1:M) {
  for (t in 1:n.occasions) {
    u[i,t] <- z[i,t]*w[i]      # Deflated latent state (u)
  }
}
for (i in 1:M) {
  recruit[i,1] <- u[i,1]
  for (t in 2:n.occasions) {
    recruit[i,t] <- (1-u[i,t-1]) * u[i,t]
  } #t
} #i

```

```

for (t in 1:n.occasions) {
  N[t] <- sum(u[1:M,t])           # Actual population size
  B[t] <- sum(recruit[1:M,t])     # Number of entries
} #t
for (i in 1:M) {
  Nind[i] <- sum(u[i,1:n.occasions])
  Nalive[i] <- 1-equals(Nind[i], 0)
} #i
Nsuper <- sum(Nalive[])           # Superpopulation size
}
",fill=TRUE)
sink()

```

Under the restricted occupancy or the multistate JS model and assuming  $T$  capture occasions and time-dependent parameters, we estimate  $T \gamma$  parameters,  $T - 1$  survival, and  $T$  capture parameters. Under the superpopulation approach, we estimate the same number of survival and capture parameters, but only  $T - 1$  entry parameters are separately estimable (note that  $b_T$  is 1 minus the sum of the other  $b$ ) plus the inclusion parameter  $\psi$ . Thus, the total number of parameters is the same in all three formulations. This illustrates the fact that the different formulations are reparameterizations of the same basic model. In Section 10.6, we provide a summary of the relationships between the quantities in each. The parameterization with the least impact of the priors can then be specified.

Naturally, all model formulations can be extended using the GLM or GLMM framework, as we did for the models in Chapters 6 and 7. Care must be taken, however, with age-dependent models (Brownie et al., 1986; Williams et al., 2002). The age of all individuals at initial capture must be known, and the entry time to the population must be somewhere between the birth of the individual and its initial capture. Capture probabilities depend on age, but the capture probability of the first age class and therefore also the population size for this age class cannot be estimated. The multistate formulation of the JS model seems to be a framework with which age-dependent models can be fitted in the Bayesian paradigm, but the problem remains that population size of the first age class cannot be estimated. Care must also be taken when entry probabilities are modeled because they need to sum to 1, and therefore not all of them can independently be modeled.

## 10.4 MODELS WITH CONSTANT SURVIVAL AND TIME-DEPENDENT ENTRY

We start with a simple example to illustrate the application of all three model formulations. We first simulate data using function `simul.js` and then analyze them. The simulation code needs the parameter of

the superpopulation formulation as input, and thus we specify the superpopulation size, entry, survival and capture probabilities, as well as the number of occasions. With  $T$  occasions,  $T$  entry probabilities must be defined, and they have to sum to 1. In the absence of any individual heterogeneity, there are  $T$  capture probabilities and  $T - 1$  survival probabilities that need to be determined.

We assume a 7-year study of survival and recruitment in black grouse (Fig. 10.3). Each spring black grouse are captured at a lek. They are marked and may be captured again in later years. We assume that annual survival is 0.7, and capture probability is 0.5; both are constant over time. Superpopulation size is 400 individuals. We assume the entry probability to be 0.11 for all occasions but the first one, when it is given by  $1 - 6 * 0.11 = 0.34$ .

```
# Define parameter values
n.occasions <- 7                                # Number of capture occasions
N <- 400                                         # Superpopulation size
phi <- rep(0.7, n.occasions-1)                   # Survival probabilities
b <- c(0.34, rep(0.11, n.occasions-1))        # Entry probabilities
p <- rep(0.5, n.occasions)                        # Capture probabilities
```



**FIGURE 10.3** Displaying black grouse cock (*Tetrao tetrix*), Finland, 2005 (Photograph by J. Peltomäki).

```

PHI <- matrix(rep(phi, (n.occasions-1)*N), ncol=n.occasions-1,
               nrow=N, byrow=T)
P <- matrix(rep(p, n.occasions*N), ncol=n.occasions, nrow=N,
               byrow=T)

# Function to simulate capture-recapture data under the JS model
simul.js <- function(PHI, P, b, N) {
  B <- rmultinom(1, N, b) # Generate no. of entering ind. per occasion
  n.occasions <- dim(PHI)[2] + 1
  CH.sur <- CH.p <- matrix(0, ncol=n.occasions, nrow=N)

  # Define a vector with the occasion of entering the population
  ent.occ <- numeric()
  for (t in 1:n.occasions) {
    ent.occ <- c(ent.occ, rep(t, B[t]))
  }

  # Simulate survival
  for (i in 1:N) {
    CH.sur[i, ent.occ[i]] <- 1    # Write 1 when ind. enters the pop.
    if (ent.occ[i] == n.occasions) next
    for (t in (ent.occ[i]+1):n.occasions) {
      # Bernoulli trial: has individual survived occasion?
      sur <- rbinom(1, 1, PHI[i,t-1])
      ifelse (sur==1, CH.sur[i,t] <- 1, break)
    } #t
  } #i

  # Simulate capture
  for (i in 1:N) {
    CH.p[i,] <- rbinom(n.occasions, 1, P[i,])
  } #i

  # Full capture-recapture matrix
  CH <- CH.sur * CH.p

  # Remove individuals never captured
  cap.sum <- rowSums(CH)
  never <- which(cap.sum == 0)
  CH <- CH[-never,]
  Nt <- colSums(CH.sur)      # Actual population size
  return(list(CH=CH, B=B, N=Nt))
}

# Execute simulation function
sim <- simul.js(PHI, P, b, N)
CH <- sim$CH

```

In our case, the resulting capture–recapture matrix has the dimension of  $7 \times 294$ , and thus 294 among the 400 individuals were captured. Besides the capture-histories, the function returns the number of individuals that newly entered the population and the actual population size at each occasion. We analyze the simulated data with all three model formulations, thus illustrating the advantages and disadvantages of each.

### 10.4.1 Analysis of the JS Model as a Restricted Occupancy Model

We first need to augment the observed capture–recapture data. We must make sure that we augment the data by a large enough number of pseudo-individuals (see Section 6.2.1).

```
# Augment the capture-histories by nz pseudo-individuals
nz <- 500
CH.aug <- rbind(CH, matrix(0, ncol = dim(CH)[2], nrow = nz))
```

Then, we define initial values and parameters we want to monitor, set the MCMC specifications, run the model and print the results.

```
# Bundle data
bugs.data <- list(y = CH.aug, n.occasions = dim(CH.aug)[2],
  M = dim(CH.aug)[1])

# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1),
  mean.p = runif(1, 0, 1), z = CH.aug)}

# Parameters monitored
parameters <- c("psi", "mean.p", "mean.phi", "b",
  "Nsuper", "N", "B", "gamma")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 11 min)
js.occ <- bugs(bugs.data, inits, parameters, "js-rest.occ.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

print(js.occ, digits = 3)

      mean      sd     2.5%    25%    50%    75%   97.5%   Rhat n.eff
psi      0.522  0.032   0.465   0.500   0.522   0.543   0.590 1.002  1800
mean.p   0.473  0.035   0.407   0.448   0.472   0.497   0.542 1.003  930
mean.phi 0.718  0.025   0.668   0.701   0.719   0.736   0.768 1.001 2400
b[1]     0.328  0.040   0.252   0.300   0.327   0.354   0.408 1.003  870
[ ... ]
b[7]     0.089  0.029   0.036   0.068   0.088   0.108   0.150 1.004 3000
Nsuper   406.364 21.013 369.975 391.000 405.000 419.000 451.000 1.001 2500
N[1]     134.027 15.971 106.000 123.000 132.000 144.000 169.000 1.003  760
[ ... ]
N[7]     150.457 14.373 126.000 140.000 149.000 159.000 183.000 1.002 1800
B[1]     134.027 15.971 106.000 123.000 132.000 144.000 169.000 1.003  760
[ ... ]
B[7]     35.751 11.451 14.975 28.000 35.000 43.000 59.000 1.001 2100
[ ... ]
```

### 10.4.2 Analysis of the JS Model as a Multistate Model

We need to add a dummy occasion before the first real occasion, augment the data set, and recode that data to match the codes of the observed states.

```
# Add dummy occasion
CH.du <- cbind(rep(0, dim(CH) [1]), CH)

# Augment data
nz <- 500
CH.ms <- rbind(CH.du, matrix(0, ncol = dim(CH.du) [2], nrow = nz))

# Recode CH matrix: a 0 is not allowed in WinBUGS!
CH.ms [CH.ms==0] <- 2                                # Not seen = 2, seen = 1
```

Then, we run the analysis.

```
# Bundle data
bugs.data <- list(y = CH.ms, n.occasions = dim(CH.ms) [2],
                  M = dim(CH.ms) [1])

# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1),
                         mean.p = runif(1, 0, 1), z = cbind(rep(NA, dim(CH.ms) [1]),
                         CH.ms [, -1]))}

# Parameters monitored
parameters <- c("mean.p", "mean.phi", "b", "Nsuper", "N", "B")

# MCMC settings
ni <- 20000
nt <- 3
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 32 min)
js.ms <- bugs(bugs.data, inits, parameters, "js-ms.bug", n.chains = nc,
               n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
               bugs.dir, working.directory = getwd())

print(js.ms, digits = 3)

      mean      sd    2.5%     25%     50%     75%   97.5%   Rhat n.eff
mean.p     0.473  0.036    0.403    0.449    0.473    0.497   0.542 1.009    360
mean.phi    0.720  0.026    0.668    0.702    0.720    0.737   0.770 1.001 15000
b[1]       0.328  0.040    0.254    0.300    0.326    0.354   0.410 1.002   1500
[ ... ]
b[7]       0.088  0.029    0.032    0.068    0.087    0.107   0.146 1.001 11000
Nsuper    405.668 19.807 370.000 392.000 405.000 419.000 447.000 1.015    170
N[1]      133.950 16.146 106.000 122.000 133.000 144.000 169.000 1.010    280
[ ... ]
N[7]      150.550 14.173 126.000 140.000 150.000 160.000 181.000 1.003    970
B[1]      133.950 16.146 106.000 122.000 133.000 144.000 169.000 1.010    280
[ ... ]
B[7]      35.503 11.277 13.000 28.000 35.000 43.000 58.000 1.001 15000
[ ... ]
```

### 10.4.3 Analysis of the JS Model Under the Superpopulation Parameterization

This analysis requires the same preparation as the restricted occupancy formulation. We augment the observed capture–recapture data and run the analysis.

```
# Augment capture-histories by nz pseudo-individuals
nz <- 500
CH.aug <- rbind(CH, matrix(0, ncol = dim(CH)[2], nrow = nz))

# Bundle data
bugs.data <- list(y = CH.aug, n.occasions = dim(CH.aug)[2],
  M = dim(CH.aug)[1])

# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1),
  mean.p = runif(1, 0, 1), psi = runif(1, 0, 1), z = CH.aug)}

# Parameters monitored
parameters <- c("psi", "mean.p", "mean.phi", "b", "Nsuper",
  "N", "B", "mu")

# MCMC settings
ni <- 5000
nt <- 3
nb <- 2000
nc <- 3

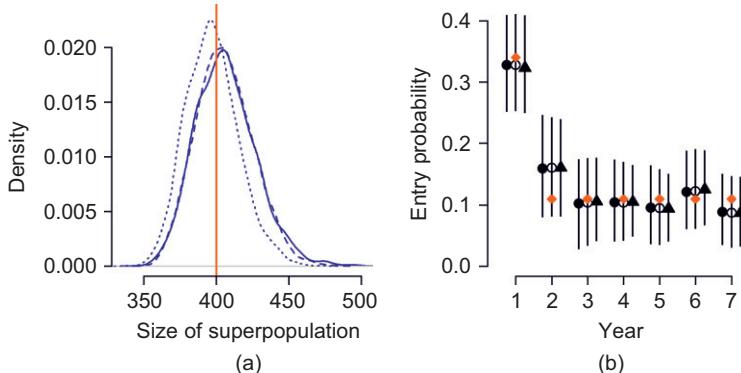
# Call WinBUGS from R (BRT 40 min)
js.super <- bugs(bugs.data, inits, parameters, "js-super.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

print(js.super, digits = 3)

      mean     sd    2.5%    25%    50%    75%   97.5% Rhat n.eff
psi      0.504  0.029   0.449   0.484   0.504   0.524   0.563 1.001  3000
mean.p   0.483  0.034   0.417   0.459   0.482   0.506   0.548 1.002  1700
mean.phi 0.720  0.026   0.671   0.702   0.719   0.737   0.773 1.002  1500
b[1]     0.323  0.040   0.251   0.295   0.320   0.348   0.408 1.017  180
[ ... ]
b[7]     0.087  0.029   0.034   0.067   0.087   0.106   0.145 1.007  300
Nsuper   396.991 18.199 364.975 384.000 396.000 408.000 436.000 1.001  3000
N[1]     129.561 15.079 103.000 119.000 128.000 139.000 161.000 1.010  240
[ ... ]
N[7]     147.403 13.196 125.000 138.000 146.000 156.000 176.000 1.001  2300
B[1]     129.561 15.079 103.000 119.000 128.000 139.000 161.000 1.010  240
[ ... ]
B[7]     34.215 10.768 14.000 27.000 34.000 41.000 56.000 1.006  380
[ ... ]
```

### 10.4.4 Comparison of Estimates

Posterior means and standard deviations are nearly identical under all three parameterizations. This is shown graphically for the superpopulation



**FIGURE 10.4** (a) Posterior distributions of the size of the number of black grouses ever alive during the study (superpopulation size,  $N_s$ ) using the restricted occupancy (solid), multistate (dashed), and superpopulation parameterizations (dotted). The vertical red line shows the data generating size of the superpopulation. (b) Estimates of annual entry probabilities under the restricted occupancy (closed circle), multistate (open circle), and superpopulation parameterization (closed triangle), as well as the data-generating parameters (red diamond). The vertical lines are 95% CRI.

size and the entry probabilities (Fig. 10.4). We would expect to get exactly the same results for the restricted occupancy and the multistate formulation, but not for the superpopulation formulation. The latter uses different priors than the two former. This may make a difference with a small data set. The three approaches differ in terms of computing time, which is much shorter for the restricted occupancy formulation. The main advantage of the multistate formulation is its flexibility to model different age classes, movement between sites or to integrate other data types such as dead recoveries. The main advantage of the superpopulation formulation is the ability to specify directly a prior for the superpopulation size and to model entry probabilities directly. Entry probabilities are parameters of direct biological interest, whereas removal entry probabilities are mere mathematical constructs. Thus, if the goal of an analysis is to understand the arrival of individuals as a function of covariates, the superpopulation formulation is the best choice. A difficulty could be that the entry probabilities must sum to 1, so not all parameters can be freely modeled. This is the same difficulty as for multistate models with movements among three or more sites (Section 9.6). In order to improve computation speed, it is again possible to provide as data information about the latent state variable  $z$  (see Section 7.3.1). We have not done this here explicitly because the specification of them and of the corresponding initial values differs for each approach. However, we make use of this information in the solutions of the exercises (Section 10.9).

```
# Code to produce Fig. 10.4
par(mfrow=c(1,2), mar=c(5, 6, 2, 1), mgp=c(3.4, 1, 0), las=1)
plot(density(js.occ$sims.list$Nsuper), main = "", xlab = "",
      ylab = "Density", frame = FALSE, lwd=2, ylim=c(0, 0.023),
      col = "blue")
points(density(js.ms$sims.list$Nsuper), type = "l", lty = 2,
      col = "blue", lwd=2)
points(density(js.super$sims.list$Nsuper), type = "l", lty = 3,
      col = "blue", lwd=2)
abline(v=N, col = "red", lwd=2)
mtext("Size of superpopulation", 1, line = 3)

b1.lower <- b2.lower <- b3.lower <- b1.upper <- b2.upper <- b3.upper <-
  numeric()
for (t in 1:n.occasions){
  b1.lower[t] <- quantile(js.occ$sims.list$b[,t], 0.025)
  b2.lower[t] <- quantile(js.ms$sims.list$b[,t], 0.025)
  b3.lower[t] <- quantile(js.super$sims.list$b[,t], 0.025)
  b1.upper[t] <- quantile(js.occ$sims.list$b[,t], 0.975)
  b2.upper[t] <- quantile(js.ms$sims.list$b[,t], 0.975)
  b3.upper[t] <- quantile(js.super$sims.list$b[,t], 0.975)
}
time <- 1:n.occasions
plot(x=time-0.25, y=js.occ$mean$b, xlab = "", ylab = "Entry
      probability", frame = FALSE, las = 1, xlim=c(0.5, 7.5), pch = 16,
      ylim=c(0, max(c(b1.upper, b2.upper))))
segments(time-0.25, b1.lower, time-0.25, b1.upper)
points(x=time, y=js.ms$mean$b, pch = 1)
segments(time, b2.lower, time, b2.upper)
points(x=time+0.25, y=js.super$mean$b, pch = 17)
segments(time+0.25, b3.lower, time+0.25, b3.upper)
points(x=time, y=b, pch = 18, col = "red")
mtext("Year", 1, line = 3)
```

## 10.5 MODELS WITH INDIVIDUAL CAPTURE HETEROGENEITY

Here, we show how an individual random effect on capture probability can be included to avoid negative bias in population size estimates (Pledger and Efford, 1998; Link, 2003; Royle and Dorazio, 2008); also see Chapter 6. We assume an 8-year study of grey-headed woodpeckers (Fig. 10.5), where birds are attracted with playback calls and caught in mist nets. There is individual variation in the aggressive reaction to the calls; hence, capture probability differs by individual. We assume a superpopulation size of 300, mean survival probability of 0.75, and annual entry probability of 0.09, except for 0.37 at the first occasion. Mean capture probability is 0.6, with an individual variance of 1 on the logit scale. We first simulate one data set and then analyze it with the superpopulation formulation.



**FIGURE 10.5** Male grey-headed woodpecker (*Picus canus*), Finland, 2002 (Photograph by T. Muukkonen).

```

# Define parameter values
n.occasions <- 8                                # Number of capture occasions
N <- 300                                         # Size of the superpopulation
phi <- rep(0.75, n.occasions-1)                  # Survival probabilities
b <- c(0.37, rep(0.09, n.occasions-1))        # Entry probabilities
mean.p <- 0.6                                     # Mean capture probability
var.p <- 1                                         # Indv. Variance of capture prob.
p <- plogis(rnorm(N, qlogis(mean.p), var.p^0.5))
PHI <- matrix(rep(phi, (n.occasions-1)*N), ncol=n.occasions-1,
               nrow=N, byrow=T)
P <- matrix(rep(p, n.occasions), ncol=n.occasions, nrow=N, byrow=F)

# Execute simulation function
sim <- simul.js(PHI, P, b, N)
CH <- sim$CH
  
```

Here is the BUGS code.

```

# Specify model in BUGS language
sink("js-super-indran.bug")
cat("
model {
```

```

# Priors and constraints
for (i in 1:M) {
  for (t in 1:(n.occasions-1)) {
    phi[i,t] <- mean.phi
  } #t
  for (t in 1:n.occasions) {
    logit(p[i,t]) <- mean.lp + epsilon[i]
  } #t
} #i

mean.phi ~ dunif(0, 1)           # Prior for mean survival
mean.lp <- log(mean.p / (1-mean.p))
mean.p ~ dunif(0, 1)             # Prior for mean capture
for (i in 1:M) {
  epsilon[i] ~ dnorm(0, tau) I(-15,15)
}
tau <- pow(sigma, -2)
sigma ~ dunif(0, 5)              # Prior for sd of indv. variation of p
sigma2 <- pow(sigma, 2)
psi ~ dunif(0, 1)                # Prior for inclusion probability

# Dirichlet prior for entry probabilities
for (t in 1:n.occasions) {
  beta[t] ~ dgamma(1, 1)
  b[t] <- beta[t] / sum(beta[1:n.occasions])
}

# Convert entry probs to conditional entry probs
nu[1] <- b[1]
for (t in 2:n.occasions) {
  nu[t] <- b[t] / (1-sum(b[1:(t-1)]))
} #t

# Likelihood
for (i in 1:M) {
  # First occasion
  # State process
  w[i] ~ dbern(psi)           # Draw latent inclusion
  z[i,1] ~ dbern(nu[1])
  # Observation process
  mu1[i] <- z[i,1] * p[i,1] * w[i]
  y[i,1] ~ dbern(mu1[i])

  # Subsequent occasions
  for (t in 2:n.occasions) {
    # State process
    q[i,t-1] <- 1-z[i,t-1]
    mu2[i,t] <- phi[i,t-1] * z[i,t-1] + nu[t] * prod(q[i,1:(t-1)])
    z[i,t] ~ dbern(mu2[i,t])
    # Observation process
    mu3[i,t] <- z[i,t] * p[i,t] * w[i]
    y[i,t] ~ dbern(mu3[i,t])
  } #t
} #i

```

```

# Calculate derived population parameters
for (i in 1:M) {
  for (t in 1:n.occasions){
    u[i,t] <- z[i,t]*w[i]          # Deflated latent state (u)
  }
}
for (i in 1:M) {
  recruit[i,1] <- u[i,1]
  for (t in 2:n.occasions){
    recruit[i,t] <- (1-u[i,t-1]) * u[i,t]
  } #t
} #i
for (t in 1:n.occasions){
  N[t] <- sum(u[1:M,t])           # Actual population size
  B[t] <- sum(recruit[1:M,t])     # Number of entries
} #t
for (i in 1:M) {
  Nind[i] <- sum(u[i,1:n.occasions])
  Nalive[i] <- 1-equals(Nind[i], 0)
} #i
Nsuper <- sum(Nalive[])           # Superpopulation size
}
",fill=TRUE)
sink()

```

Finally, we augment the data set and run the analysis in BUGS.

```

# Augment the capture-histories by nz pseudo-individuals
nz <- 300
CH.aug <- rbind(CH, matrix(0, ncol = dim(CH)[2], nrow = nz))

# Bundle data
bugs.data <- list(y = CH.aug, n.occasions = dim(CH.aug)[2],
  M = dim(CH.aug)[1])

# Initial values
inits <- function() {list(mean.phi = runif(1, 0, 1),
  mean.p = runif(1, 0, 1), sigma = runif(1, 0, 1), z = CH.aug)}

# Parameters monitored
parameters <- c("sigma2", "psi", "mean.p", "mean.phi", "N", "Nsuper",
  "b", "B")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 179 min)
js.ran <- bugs(bugs.data, inits, parameters, "js-super-indran.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

```
print(js.ran, digits = 3)

      mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
sigma2    0.191  0.281  0.001  0.019  0.083  0.242  0.980 1.135    23
psi       0.542  0.033  0.483  0.520  0.540  0.562  0.613 1.002   2000
mean.p    0.629  0.037  0.550  0.607  0.631  0.655  0.696 1.003   1100
mean.phi   0.746  0.021  0.704  0.732  0.746  0.760  0.787 1.005   520
N[1]    121.716 10.153 104.000 115.000 121.000 128.000 143.000 1.005   520
[ ... ]
N[8]    113.286  9.498  97.000 107.000 113.000 119.000 134.000 1.005   490
Nsuper  291.239 13.436 270.000 282.000 290.000 298.000 323.000 1.003   780
b[1]     0.410  0.039  0.335  0.383  0.409  0.436  0.485 1.003   770
[ ... ]
b[8]     0.155  0.030  0.099  0.135  0.154  0.175  0.215 1.003   800
B[1]    121.716 10.153 104.000 115.000 121.000 128.000 143.000 1.005   520
[ ... ]
B[8]    45.505  7.246  32.000  40.000  45.000  50.000  61.000 1.002  1200
[ ... ]
```

The posterior means match up quite well with the data-generating parameters. Survival, capture and the entry probabilities are estimated well, whereas the estimate for the individual variation is lower than the data-generating parameter. Yet, as always, in order to study whether the model produces unbiased estimates, the exercise would have to be repeated many times and the average behavior of estimates studied.

## 10.6 CONNECTIONS BETWEEN PARAMETERS, FURTHER QUANTITIES AND SOME REMARKS ON IDENTIFIABILITY

All three approaches to the JS model under data augmentation are related. The restricted occupancy and the multistate formulation are exactly equivalent in terms of parameterization and priors. In contrast, the superpopulation formulation has a different parameterization (in terms of  $b$  and  $\gamma$  instead of  $\gamma$ ) and consequently needs priors for other parameters. Here, we summarize connections between different parameters in the three approaches and also their relation to further quantities of interest.

The expected number of newly entered individuals per occasion is

$$\begin{aligned} E(B_1) &= M\gamma_1 = N_s b_1 \\ E(B_2) &= M(1 - \gamma_1)\gamma_2 = N_s b_2 \\ &\dots \\ E(B_t) &= M \prod_{i=1}^{t-1} (1 - \gamma_i) \gamma_t = N_s b_t \end{aligned}$$

Let us denote the probability that an “individual” within the augmented data  $M$  is a member of the true individuals  $N_s$  with  $\psi = N_s/M$ . After some algebra, we see that

$$\gamma_1 = \psi b_1, \quad \gamma_2 = \psi \frac{b_2}{1 - b_1}, \quad \dots, \quad \gamma_t = \psi \frac{b_t}{1 - \sum_{i=1}^{t-1} b_i}$$

Likewise, we can calculate  $b$  from  $\gamma$  as

$$b_1 = \frac{1}{\psi} \gamma_1, \quad b_2 = \frac{1}{\psi} (1 - \gamma_1) \gamma_2, \dots$$

Because all  $b$  sum to 1,  $b_T$  at the last occasion  $T$  is

$$b_T = \frac{1}{\psi} \gamma_T \prod_{i=1}^{T-1} (1 - \gamma_i) = 1 - \frac{1}{\psi} \left( \gamma_1 + \sum_{i=1}^{T-1} \left( \gamma_{i+1} \prod_{j=i+1}^T (1 - \gamma_j) \right) \right)$$

Therefore, we can directly calculate  $\psi$  from  $\gamma$  as

$$\psi = 1 - \prod_{i=1}^T (1 - \gamma_i)$$

Pradel (1996) and Link and Barker (2005) used a further parameterization to model the recruitment process. Instead of an entry probability that refers either to the size of the augmented data set ( $\gamma$ , restricted occupancy parameterization and multistate model) or to the size of the superpopulation ( $b$ , superpopulation parameterization), they defined a per-capita entry probability ( $f$ ). This quantity is computed as

$$f_t = \frac{B_t}{N_t}$$

and expresses the fraction of new individuals at  $t$  per individual alive at  $t$ . Expressing recruitment in this way results in the biologically most meaningful quantity. For the three models in this chapter, the per-capita entry probability can easily be estimated as a derived parameter, but to model this quantity directly, a different model parameterization is needed (Link and Barker, 2010).

The population growth rate ( $\lambda$ ) is easily computed as a derived quantity from the estimated population sizes or survival and per-capita entry probability:

$$\lambda_t = \frac{N_{t+1}}{N_t} = \phi_t + f_t.$$

As almost always with complex models, not all parameters may be separately identifiable. It is well known that some parameters are not identifiable in a JS model with a time-dependent structure on survival,

capture, and entry probabilities: the first entry and capture probabilities and the last survival and capture probabilities, respectively, are only estimable as products (i.e.,  $b_1 p_1$ ,  $\phi_{T-1} p_T$ ; Schwarz and Arnason, 1996). This confounding occurs for all three parameterizations in this chapter. To verify parameter estimability, we can inspect the prior/posterior overlap (see Section 7.9) or see whether an analysis of simulated data yields estimates that resemble the input parameters.

## 10.7 ANALYSIS OF A REAL DATA SET: SURVIVAL, RECRUITMENT AND POPULATION SIZE OF LEISLER'S BATS

---

As a real-world example, we reanalyze capture–recapture data from adult female Leisler's bat (Fig. 7.11) from Section 7.11 (see also Schorcht et al., 2009) under the JS model. We only use a homogenous subset of these data consisting of locally born adult females. We are interested in population size and in the variability of local recruitment over time. Our data set consists of 181 individuals. Initial modeling suggested that adult survival was subject to strong temporal variation, whereas recapture probability was constant over time. We therefore fit a model with temporal random effects in survival, fixed time effects in recruitment, and a constant capture rate. We denote this model as  $(\phi_t, b_t, p)$ .

We first specify the BUGS model. We use the restricted occupancy formulation because of its advantage in terms of computation speed. The model code needs an adaptation: annual survival is now specified as a random effect.

```
# Specify model in BUGS language
sink("js-temporal.bug")
cat("
model {
  # Priors and constraints
  for (i in 1:M) {
    for (t in 1:(n.occasions-1)) {
      logit(phi[i,t]) <- mean.lphi + epsilon[t]
    } #
    for (t in 1:n.occasions) {
      p[i,t] <- mean.p
    } #
  } #
  mean.p ~ dunif(0, 1)          # Prior for mean capture
  mean.phi ~ dunif(0, 1)         # Prior for mean survival
  mean.lphi <- log(mean.phi / (1-mean.phi))
  for (t in 1:(n.occasions-1)) {
    epsilon[t] ~ dnorm(0, tau)
  }
}
```

```

tau <- pow(sigma, -2)
sigma ~ dunif(0, 5)           # Prior for sd of indv. variation of phi
sigma2 <- pow(sigma, 2)

for (t in 1:n.occasions) {
  gamma[t] ~ dunif(0, 1)
} #t

# Likelihood
for (i in 1:M) {
  # First occasion
  # State process
  z[i,1] ~ dbern(gamma[1])
  mu1[i] <- z[i,1] * p[i,1]
  # Observation process
  y[i,1] ~ dbern(mu1[i])

  # Subsequent occasions
  for (t in 2:n.occasions) {
    # State process
    q[i,t-1] <- 1-z[i,t-1]
    mu2[i,t] <- phi[i,t-1] * z[i,t-1] + gamma[t] * prod(q[i,1:(t-1)])
    z[i,t] ~ dbern(mu2[i,t])
    # Observation process
    mu3[i,t] <- z[i,t] * p[i,t]
    y[i,t] ~ dbern(mu3[i,t])
  } #t
} #i

# Calculate derived population parameters
for (t in 1:n.occasions) {
  qgamma[t] <- 1-gamma[t]
}
cprob[1] <- gamma[1]
for (t in 2:n.occasions) {
  cprob[t] <- gamma[t] * prod(qgamma[1:(t-1)])
} #t
psi <- sum(cprob[])           # Inclusion probability
for (t in 1:n.occasions) {
  b[t] <- cprob[t] / psi      # Entry probability
} #t
for (i in 1:M) {
  recruit[i,1] <- z[i,1]
  for (t in 2:n.occasions) {
    recruit[i,t] <- (1-z[i,t-1]) * z[i,t]
  } #t
} #i
for (t in 1:n.occasions) {
  N[t] <- sum(z[1:M,t])       # Actual population size
  B[t] <- sum(recruit[1:M,t]) # Number of entries
} #t
for (i in 1:M) {
  Nind[i] <- sum(z[i,1:n.occasions])
  Nalive[i] <- 1-equals(Nind[i], 0)
} #i

```

```
Nsuper <- sum(Nalive[])
# Size of superpopulation
}
",fill=TRUE)
sink()
```

Next, we load the capture–recapture data and augment the data.

```
leis <- as.matrix(read.table("leisleri.txt", sep = " ",
header = FALSE))
nz <- 300
CH.aug <- rbind(leis, matrix(0, ncol = dim(leis)[2], nrow = nz))
```

Then, we run the analysis.

```
# Bundle data
bugs.data <- list(y = CH.aug, n.occasions = dim(CH.aug)[2],
M = dim(CH.aug)[1])

# Initial values
inits <- function() {list(mean.phi = runif(1, 0, 1),
mean.p = runif(1, 0, 1), sigma = runif(1, 0, 1), z = CH.aug)}

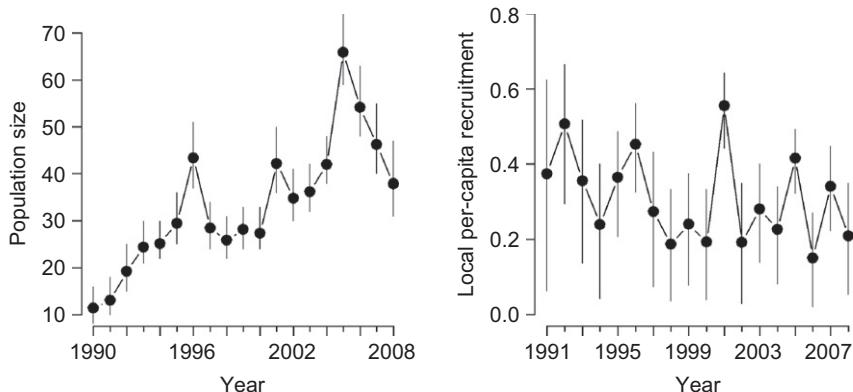
# Parameters monitored
parameters <- c("psi", "mean.p", "sigma2", "mean.phi", "N",
"Nsuper",
"b", "B")

# MCMC settings
ni <- 10000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 127 min)
nl <- bugs(bugs.data, inits, parameters, "js-tempran.bug", n.chains =
  nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

print(nl, digits = 3)
      mean     sd    2.5%    25%    50%    75%   97.5%   Rhat n.eff
mean.p    0.725  0.030   0.665   0.704   0.726   0.746   0.782 1.001  2500
sigma2    0.477  0.350   0.064   0.247   0.397   0.613   1.363 1.012  260
mean.phi   0.743  0.039   0.669   0.717   0.743   0.768   0.821 1.003  690
[ ... ]
Nsuper    205.896  6.576 194.000 201.000 205.000 210.000 220.000 1.003  2500
[ ... ]
```

The model runs quite slowly, but relatively few MCMC samples are sufficient to obtain convergence. Estimates of mean survival and its temporal variability are close to those obtained under the CJS model (see Section 7.11), and also the capture probabilities differ only very slightly (JS: 0.725 vs. CJS: 0.747). Such a difference is not unusual since the parameters are not exactly the same: in the CJS model, it is the recapture probability, whereas in the JS, it is the capture probability for all occasions, including the first. The estimated population sizes suggest that bat



**FIGURE 10.6** Posterior means of population sizes and of per-capita recruitment of adult female Leisler's bats. The vertical lines are 95% CRI.

population increased until 2005 and declined afterwards (Fig. 10.6). The per-capita recruitment seemed to have declined slightly over time, but showed strong annual fluctuations (Fig. 10.6). Note that the per-capita recruitment refers to local recruitment only because the data set consisted only of locally born bats.

```
# Code to produce Fig. 10.6
# Calculate per-capita recruitment
T <- dim(leis)[2]
f <- matrix(NA, ncol=T, nrow = length(nl$sims.list$B[,1]))
for (t in 1:(T-1)){
  f[,t] <- nl$sims.list$B[,t+1] / nl$sims.list$N[,t+1]
}
n.lower <- n.upper <- f.lower <- f.upper <- f.mean <- numeric()
for (t in 1:T){
  n.lower[t] <- quantile(nl$sims.list$N[,t], 0.025)
  n.upper[t] <- quantile(nl$sims.list$N[,t], 0.975)
}
for (t in 1:(T-1)){
  f.lower[t] <- quantile(f[,t], 0.025)
  f.upper[t] <- quantile(f[,t], 0.975)
  f.mean[t] <- mean(f[,t])
}
par(mfrow=c(1, 2))
plot(nl$mean$N, type = "b", pch = 19, ylab = "Population size", xlab = "",
     axes = F, cex = 1.5, ylim = c(10, max(n.upper)))
axis(1, at = seq(1, T, 2), labels = seq(1990, 2008, 2))
axis(1, at = 1:T, labels = rep("", T), tcl = -0.25)
axis(2, las = 1)
segments(1:T, n.lower, 1:T, n.upper)
plot(f.mean, type = "b", pch = 19, ylab = "Local per capita recruitment",
     xlab = "", axes = F, cex = 1.5, ylim = c(0, 0.8))
```

```

axis(1, at = seq(1, (T-1), 2), labels = seq(1991, 2008, 2))
axis(1, at = 1:(T-1), labels = rep("", T-1), tcl = -0.25)
axis(2, las = 1)
segments(1:(T-1), f.lower, 1:(T-1), f.upper)

```

## 10.8 SUMMARY AND OUTLOOK

We have described the Jolly-Seber (JS) model to estimate population size, survival and recruitment from capture–recapture data. The main difference to the CJS model (Chapter 7) is that the JS does not condition on first capture, that is, leading zeros before initial capture and the initial capture are modeled as well. There are several variants (parameterizations) of JS models that differ in the way how recruitment-related parameters are defined and estimated. We have illustrated three different parameterizations, showed advantages and disadvantages, as well as the connection between the parameters in them. The formulation of the JS model as a restricted occupancy model has advantages of relatively fast speed and a simple model code. The superpopulation formulation runs relatively slowly, but it is possible to directly model the entry probability and to specify a natural prior for the size of the superpopulation. We also showed that the JS model can be formulated as a multistate model. This formulation is appealing because it allows extensions in many directions. For all three formulations, we use parameter-expanded data augmentation (Royle et al., 2007a; Royle and Dorazio, 2011).

The GLM concept and random effects can be applied to all structural parameters in all three formulations of the JS model. Particular care must be taken, however, with age-dependent models and when the entry process is modeled.

Sometimes capture–recapture data arise under the robust design, that is, under a two-stage temporal sampling scheme, where the population is assumed open between primary occasions but closed between secondary, within-primary, occasions (Pollock, 1982; Kendall et al., 1997). The multi-season metapopulation estimation models in Chapters 12 and 13 also require the robust design data format. The JS model to analyze capture–recapture data sampled under the robust design is a combination of open- and closed-population models and has many advantages. First, it allows the estimation of survival, recruitment-related parameters, and population sizes as does the traditional JS model. Yet, because there is more information about the capture process, the robust design model can deal with certain violations of the basic JS model assumptions, such as permanent trap response. It also provides the basis to decompose entry probability into *in situ* recruitment and immigration (Nichols and Pollock, 1990) and to estimate population size of all age classes. The restricted occupancy and the superpopulation formulations of the JS model only require a small

modification in the observation model to fit the robust design model. In its simplest version, we just need to model the observation with the Binomial instead of the Bernoulli distribution

$$Y_{i,t} | z_{i,t} \sim \text{Binomial}(K, z_{i,t} p_{i,t})$$

where the binomial total ( $K$ ) is the number of secondary occasions (Royle and Dorazio, 2008; Royle and Dorazio, 2011). The observed data for individual  $i$  and each primary occasion  $t$  ( $Y_{i,t}$ ) is then the number of times individual  $i$  was captured during the primary occasion  $t$ . Of course, more complex models with binary response require more adaptations.

## 10.9 EXERCISES

---

1. Simulate capture–recapture data of a species for males and females. The study is conducted for 8 years; the mean survival of males is 0.75 and of females 0.5, and capture is 0.4 for both. The entry probability after the first occasion is 0.1 in both sexes. The size of the superpopulation is 300 in both sexes. Simulate one data set and analyze it with the model  $(\phi_{\text{sex}}, b_t, p)$ .
2. Simulate capture–recapture data of a species collected over 7 years. Mean survival is 0.5, mean capture is 0.6, and entry probability is 0.1 for all but the first occasion. The size of the superpopulation is assumed to be 500. Analyze the data with the model that explicitly uses constant entry probability for all occasions, but the first.
3. Simulate data for a species for which capture–recapture data are sampled and recoveries of dead individuals are available. The study runs for 10 years, mean survival is 0.5, mean capture is 0.6, mean recovery is 0.2, and entry probability is 0.1 for all but the first occasion. The size of the superpopulation is assumed to be 500. Analyze the simulated data with an appropriate model.

## 11

# Estimation of Demographic Rates, Population Size, and Projection Matrices from Multiple Data Types Using Integrated Population Models

## OUTLINE

11.1 Introduction	348
11.2 Developing an Integrated Population Model (IPM)	350
11.2.1 First Step: Define the Link between Changes in Population Size and Demographic Rates	350
11.2.2 Second Step: Define the Likelihoods of Each Individual Data Set	352
11.2.3 Third Step: Formulate the Joint Likelihood	354
11.3 Example of a Simple IPM (Counts, Capture–Recapture, Reproduction)	357
11.3.1 Load Data	357
11.3.2 Analysis of the Model	358
11.4 Another Example of an IPM: Estimating Productivity without Explicit Productivity Data	363
11.5 IPMs for Population Viability Analysis	366
11.6 Real Data Example: Hoopoe Population Dynamics	371
11.7 Summary and Outlook	379
11.8 Exercises	380

## 11.1 INTRODUCTION

Chapters 6–10 introduced models to separately estimate population size, recruitment, survival, or movement probabilities from various types of data. Often, studies focus on population dynamics, that is, changes in abundance over time and demographic causes of those changes. In that case, the obvious thing to do is to combine different available data sets to get deeper insights into population dynamics and better estimates of the demographic quantities. The link between population size and demographic, or vital, rates is straightforward because the change in population size over time is the direct result of these demographic rates. More formally, we have

$$N_{t+1} = N_t \times g(s, f)$$

where  $N_t$  is population size in year  $t$ ,  $s$  is survival,  $f$  is productivity, and  $g$  is some function. When a study population is geographically open, the argument of  $g$  also includes terms for immigration and emigration. Looking at this link between population size and demographic rates, an important point becomes evident: time-series data on the size of a population contain information about the underlying demographic processes. When data on population size and demographic rates are analyzed jointly, three benefits accrue:

1. There is information about demographic rates both from explicit data on demographic rates (e.g., mark-recoveries for survival) and from data on population size. As a result, demographic rates are estimated with increased precision in a combined analysis. At the same time, changes in population size are the result of four demographic rates, and incorporation of data on demographic rates will likely improve population size estimates.
2. When explicit data about a demographic rate (such as productivity) is lacking, it may be possible to estimate the demographic rate by exploiting the information about it from the data on population size.
3. A combined analysis of data on demographic rates and population size allows the simultaneous study of population processes (demographic rates) and the result of these processes (population size). This can be important as we are often interested in the demographic causes for population change in population ecology, conservation biology, or wildlife management. Thus, a joint analysis allows a comprehensive assessment of the state and the dynamics of a population (Baillie, 1991).

A relatively new modeling framework, which uses different types of data to simultaneously estimate trajectories of population size and demographic

parameters, is usually called an integrated population model (Besbeas et al., 2002; reviewed by Schaub and Abadi, 2011). This analytical framework uses a population model for the time-series data on population size and combines it with additional data to inform certain elements of the population model, such as survival, fecundity, or dispersal. The use of integrated population models in animal population ecology is relatively recent (Besbeas et al., 2002, 2003; Brooks et al., 2004; Thomas et al., 2005; Schaub et al., 2007; Baillie et al., 2009; Borysiewicz et al., 2009; King et al., 2010) although some varieties of these models have been used in fisheries (e.g., Elliott and Little, 2000; Maunder, 2004). Here, we focus on the Bayesian analysis of these models, although frequentist analyses using maximum likelihood are also possible and historically came before Bayesian analyses (see, e.g., Besbeas et al., 2002, 2003, 2005; Besbeas and Freeman, 2006; Gauthier et al., 2007; Tavecchia et al., 2009; Péron et al., 2010; De Valpine, 2011). Most frequentist analyses use Kalman filter techniques (Harvey, 1989) with the advantage that numerical optimization of the likelihood function is faster than posterior sampling by MCMC. Furthermore, model selection using AIC is straightforward when using maximum likelihood; however, this comes at the cost of stronger assumptions about distributional forms and linearity of the model (Brooks et al., 2004; De Valpine, 2011).

Developing an integrated population model involves three basic steps (Schaub and Abadi, 2011). First, we need to develop a population model that links the demographic rates with changes in population size. Typically, an age- or stage-classified matrix population model (Caswell, 2001) is used. Second, we write down the likelihood of all data sets available. One data set that is always required for a classical integrated population model is a time series of estimated population sizes or of population indices (counts). To separate process variability from observation errors in these data, we can use a state-space model as used in Chapter 5. Other data sets may be capture–recapture data, for which we may adopt the CJS model, or ring-recovery data. Third, we construct the joint likelihood—the likelihood of the complete model—and make inferences. Under the assumption of independence of all data sets, the joint likelihood is the product of the likelihoods of the individual data sets. In the frequentist framework, the joint likelihood would be maximized, while in a Bayesian analysis of the model, we combine it with prior distributions for all unknowns and use MCMC to sample the joint posterior distribution. To see more clearly how such a model is constructed and analyzed, we will walk through an example and comment each step.

We first use a simulated data set. We will not show or explain the simulation code, as this would take too much room. However, you can find R code for the simulation of our data in Web appendix 2, and comments on simulating these data are given in the study by Abadi et al. (2010a). Our data mimic the dynamics of a population of ortolan buntings (Fig. 11.1), a small passerine



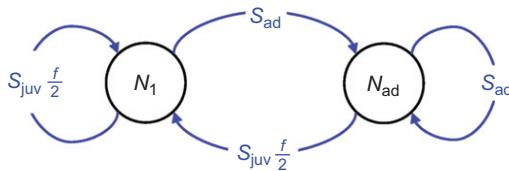
**FIGURE 11.1** Singing male ortolan bunting (*Emberiza hortulana*), Switzerland (Photograph by P. Keusch).

species. We assume a 10-year study in a large study area, where the population is closely surveyed. Each year we record the number of singing males and the number of fledglings produced in nests that are found. Both nestlings and adults are caught and ringed. Thus, we have three data sets that contain information about the dynamics of this population: population size (the number of singing males), fecundity (the total number of fledglings produced in the surveyed broods), and survival (capture–recapture data from juveniles and adults). We chose the following parameter values for the simulation of the data sets: 0.26 for juvenile survival, 0.5 for adult survival, 0.6 for recapture, and 4.0 for productivity. We assumed constant parameters over time, which results in a population growth rate of 1.02.

## **11.2 DEVELOPING AN INTEGRATED POPULATION MODEL (IPM)**

### **11.2.1 First Step: Define the Link between Changes in Population Size and Demographic Rates**

First, we need to link changes in population size with the demographic parameters. We use a simple female-based, age-classified population projection matrix model (Caswell, 2001). We assume that all individuals start



**FIGURE 11.2** Life-cycle graph of an ortolan bunting population. The nodes show two age classes ( $N_1$ : 1-year-old females;  $N_{\text{ad}}$ : females older than 1 year) and the arrows the transition probabilities based on the vital rates ( $S_{\text{juv}}$ : juvenile survival;  $S_{\text{ad}}$ : adult survival;  $f$ : productivity).

to reproduce at the age of 1 year and distinguish two age classes: 1-year-old individuals (1) and individuals older than one year (ad). Each year we survey the population immediately before the young are born, hence, we assume a prebreeding census. A life-cycle graph of the model is shown in Fig. 11.2. A mathematical description of a deterministic version of the model is

$$\begin{bmatrix} N_1 \\ N_{\text{ad}} \end{bmatrix}_{t+1} = \begin{bmatrix} S_{\text{juv}} \frac{f}{2} & S_{\text{juv}} \frac{f}{2} \\ S_{\text{ad}} & S_{\text{ad}} \end{bmatrix} \begin{bmatrix} N_1 \\ N_{\text{ad}} \end{bmatrix}_t,$$

where  $S_{\text{juv}}$  and  $S_{\text{ad}}$  are the annual survival probabilities of juvenile and adult females, respectively,  $f$  is the number of offspring produced per female, and  $N_1$  and  $N_{\text{ad}}$  are the number of 1-year-old and adult females in the population. Fecundity is divided by 2, reflecting our assumption of an even sex ratio and because our model only keeps track of females. Care must be taken with the time indices of the demographic rates. When using a prebreeding census, as here, the projection matrix is parameterized with survival from year  $t$  to year  $t + 1$  and with fecundity in year  $t$ . With a postbreeding census, the matrix would still contain survival from year  $t$  to  $t + 1$  but fecundity in year  $t + 1$  and also a different parameterization of the projection matrix.

Another way to write the above model is in terms of the expected numbers of 1-year-old and adult individuals, respectively:

$$E(N_{1,t+1} | N_{1,t}, N_{\text{ad},t}) = N_{1,t} S_{\text{juv},t} \frac{f_t}{2} + N_{\text{ad},t} S_{\text{ad},t} \frac{f_t}{2}$$

$$E(N_{\text{ad},t+1} | N_{\text{ad},t}) = N_{1,t} S_{\text{ad},t} + N_{\text{ad},t} S_{\text{ad},t}$$

These are the expected numbers; now we want to write these equations in such a way that all relevant sources of uncertainty are included. One form of variability that is always present is demographic stochasticity. Demographic stochasticity is particularly important when population size is small (Lande, 2002). To include demographic stochasticity, we use

appropriate distributions to describe the number of individuals in year  $t + 1$ . To estimate the number of 1-year-old individuals, we must find a distribution that yields an integer value between 0 (if reproduction is 0 or all individuals die) and some big number (if reproduction is great and many survive) and whose expected value is  $N_{1,t}S_{\text{juv},t}f_t/2 + N_{\text{ad},t}S_{\text{ad},t}f_t/2$ . The Poisson distribution is an appropriate candidate:

$$N_{1,t+1} \sim \text{Poisson}\left(N_{1,t}S_{\text{juv},t}\frac{f_t}{2} + N_{\text{ad},t}S_{\text{ad},t}\frac{f_t}{2}\right).$$

For the adults, we must find a distribution that generates an integer value between 0 (if no individual survives) and  $N_{\text{ad},t}$  (if all individuals survive) and whose expected value is  $N_{1,t}S_{\text{ad},t} + N_{\text{ad},t}S_{\text{ad},t}$ . Here, the binomial distribution is appropriate for modeling such bounded counts:

$$N_{\text{ad},t+1} \sim \text{Binomial}(N_{1,t} + N_{\text{ad},t}, S_{\text{ad},t}).$$

By relating the adult and juvenile population sizes between successive years among each other in a stochastic manner, we account for demographic stochasticity.

### 11.2.2 Second Step: Define the Likelihoods of Each Individual Data Set

Now we have defined the link between the population size and the demographic parameters. The next step is to write the likelihood of all available data sets. Here we have three different data sets, and we start with the likelihood of population counts.

#### **Likelihood of the Population Count Data**

A powerful way to model population counts is a state-space model (Chapter 5). The state process describes the true but unknown population trajectory under the model defined in step 1, and the observation process links the observed population counts to the true population sizes by allowing for observation error. Thus, the state-process model is defined by the two equations developed in [Section 11.2.1](#):

$$N_{1,t+1} \sim \text{Poisson}\left(N_{1,t}S_{\text{juv},t}\frac{f_t}{2} + N_{\text{ad},t}S_{\text{ad},t}\frac{f_t}{2}\right)$$

$$N_{\text{ad},t+1} \sim \text{Binomial}(N_{1,t} + N_{\text{ad},t}, S_{\text{ad},t}).$$

The age-specific number of individuals present in the first study year must be estimated from the data (see below). There are different possibilities to write the observation model that differ in the assumed distribution of the

observation errors. Often, a normal error assumption is made; thus, the count data ( $y$ ) are modeled as

$$\begin{aligned} y_t &= (N_{1,t} + N_{\text{ad},t}) + \eta_t \\ \eta_t &\sim \text{Normal}(0, \sigma_y^2) \end{aligned}$$

where  $\sigma_y^2$  is the observation error. Another possibility is to adopt a Poisson distribution for the observation error in population counts:

$$y_t \sim \text{Poisson}(N_{1,t} + N_{\text{ad},t}).$$

A feature of the Poisson distribution is that the variance is equal to its mean, implying that the observation error increases with the population size. A third possibility is a log-normal distribution:

$$\begin{aligned} \log(y_t) &= \log(N_{1,t} + N_{\text{ad},t}) + \eta_t \\ \eta_t &\sim \text{Normal}(0, \sigma_y^2). \end{aligned}$$

Modeling the log population counts ensures that the observation error increases with population size and allows for counts that are skewed on the arithmetic scale.

Some comments are in order on the observation model. First, in our experience, the choice of the observation model (i.e., between the three variants given above) often has no strong effect on the parameter estimates (but see Knappe et al., 2011). Second, all the models can only adjust for random observation errors, that is, some sort of binomial sampling error as shown in Section 5.3. Any systematic patterns such as a trend (e.g., when observers become better over time) cannot be properly accounted for by this type of model. The general discussion about state-space models for inference about population size (see Chapter 5) also applies for this component of integrated models. Third, none of the above-mentioned observation models allows estimation of detection probability, and, consequently, the true population size remains unknown. So if the average detection probability of an individual is 0.9, then the estimated population size ( $N$ ) will on average be 0.9 times the true population size. If we want to estimate detection probability, different survey protocols must be used (see Chapters 6, 10, 12, and 13). If detection probability is stationary (i.e., fluctuates around a constant mean), this caveat does not apply for conclusions about population dynamics and for the estimation of the demographic parameters. Fourth, the survey may be restricted to certain age classes or other segments of the population or it may not differentiate between age classes, and yet, we are usually still able to estimate age-specific population sizes (see also Link et al., 2003, for an impressive example of this, although not in the context of an integrated population model). Finally, it may be possible to get separate counts of different age classes, with the advantage of being able to

extract more detailed information from the counts (Tavecchia et al., 2009). The observation equation then needs a slight adaptation.

In what follows, we will use the normal approximation for the observation error. The likelihood of the state-space model is the product of the likelihood of the observation and the process equations:

$$L_{\text{SS}}(\mathbf{y} | \mathbf{N}, \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{f}, \sigma_y^2) = L_O(\mathbf{y} | \mathbf{N}, \sigma_y^2) \times L_S(\mathbf{N} | \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{f}).$$

The state-space likelihood already contains all parameters that we would like to estimate. So one might wonder why not just use the state-space model alone to estimate these parameters; that is, why aren't the count data enough? The reason is that the parameters in this model would not be identifiable, unless we impose strong constraints on the parameters (King et al., 2010), use informative prior distributions (Thomas et al., 2005; Newman et al., 2006; Buckland et al., 2007), or have counts from several stage/age classes (Link et al., 2003; David et al., 2010). Otherwise, the parameters would not be identifiable. However, if we add more (independent) information about some or all of the parameters, the model typically becomes identifiable. Therefore, we have to go on with the definition of likelihoods of other kinds of data.

### **Likelihood of the Capture–Recapture Data**

For the capture–recapture data, we can use the likelihood of the CJS model introduced in Section 7.9, that is, the multinomial likelihood,  $L_{\text{CJS}}(\mathbf{m} | \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{p})$ . Whenever possible, we recommend the multinomial likelihood to estimate survival, because of the resulting computational benefits compared with the state-space likelihood. It requires that the capture–recapture data are summarized in the m-array format ( $\mathbf{m}$ ).

### **Likelihood of Reproductive Success Data**

We use a simple Poisson regression to model productivity (see Chapters 3 and 4). We assume that the total number of nestlings counted in year  $t$ , ( $J_t$ ), follows a Poisson distribution with parameters that are the product of the number of surveyed broods ( $R_t$ ) and productivity ( $f_t$ ), hence,  $J_t \sim \text{Poisson}(R_t f_t)$ . Thus, the likelihood is  $L_P(\mathbf{J}, \mathbf{R} | \mathbf{f})$ .

#### **11.2.3 Third Step: Formulate the Joint Likelihood**

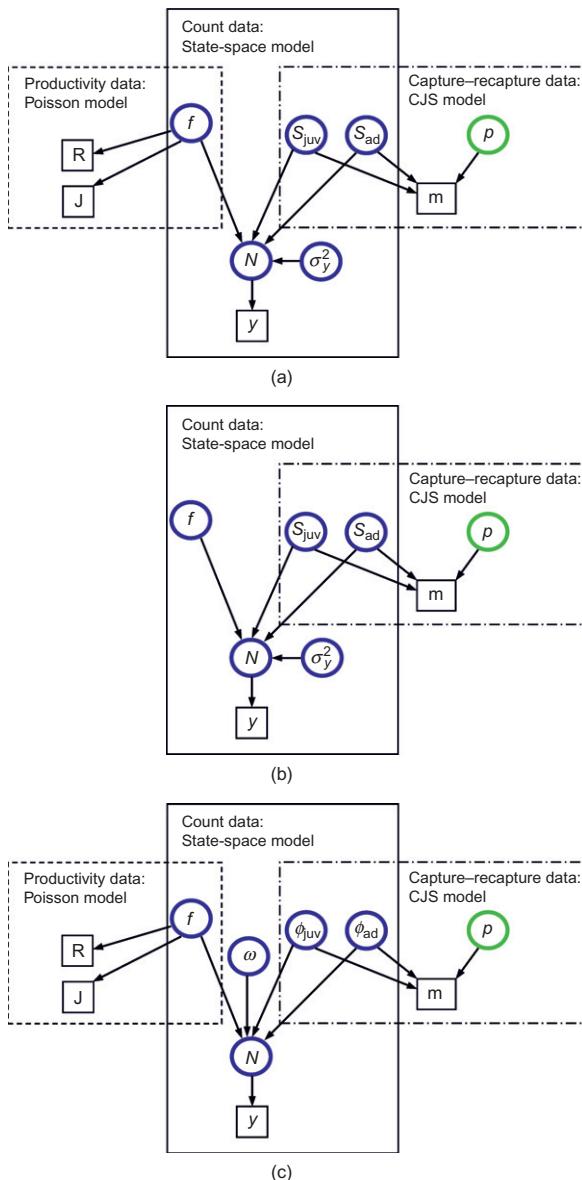
The last step is to formulate the likelihood of the complete model. This is in fact simple, if we can assume independence among the component data sets of the integrated analysis. Then, the joint likelihood is just the product of the component likelihoods:

$$\begin{aligned} L_{\text{IPM}}(\mathbf{y}, \mathbf{m}, \mathbf{J}, \mathbf{R} | \mathbf{N}, \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{f}, \mathbf{p}, \sigma_y^2) &= L_O(\mathbf{y} | \mathbf{N}, \sigma_y^2) \times L_S(\mathbf{N} | \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{f}) \\ &\quad \times L_{\text{CJS}}(\mathbf{m} | \mathbf{S}_{\text{juv}}, \mathbf{S}_{\text{ad}}, \mathbf{p}) \times L_P(\mathbf{J}, \mathbf{R} | \mathbf{f}) \end{aligned}$$

A graph of this model is shown in Fig. 11.3a. The independence of the data sets is a crucial assumption. Under one—restrictive—view, this would mean that no animal in the population count data must occur in the capture–recapture data or in the productivity data. This could be achieved if different (sub)populations are sampled: in one population we collect population count data, in another the capture–recapture data, and in the third productivity data. Alternatively, we could sample a single very large population (e.g., living in a whole country) where it is unlikely that the same individual appears in more than one data set. Although the independence assumption may then hold, we would need a new assumption that the dynamics as well as the link between population size and demography is identical in all sampled (sub)populations or across the large spatial scale. This may well be an unrealistic assumption too.

In practice, we often have different kinds of demographic data from a single and rather small population. After all, it is for small populations with limited data where the combination of multiple data sets is particularly fruitful in terms of the increased precision of parameter estimates (Schaub et al., 2007). Therefore, it is important to understand whether the violation of the independence assumption has a strong impact on the parameter estimates. Abadi et al. (2010a) have recently studied this issue. In principle, if the data sets are not independent, we would not expect biased parameter estimates, but spuriously high precision of the estimates. This is a classical result in statistics when nonindependent data are analyzed as if they were independent: the information from one individual is used multiple times, and hence, the genuine sample size is not as large as it seems. For example, if an individual is included in the population count data, it contributes to the estimation of survival in the state-space likelihood. The same individual might be included in the capture–recapture data and thus again provide information about survival via the multinomial likelihood of the CJS model. The result of this is a kind of overdispersion due to lack of independence.

Abadi et al. (2010a) simulated data types similar to the ones we use here with different degrees of nonindependence. They found that violation of the independence assumption had almost no effect on the accuracy of the parameter estimates. In contrast, Besbeas et al. (2009) found that the violation of the independence assumption had an effect when population count and mark-recovery data were combined. These divergent conclusions likely stem from the fact that different data sets contribute differently to the joint likelihood. In fact, in these classical applications, violation of independence can only occur between the state-space likelihood and another likelihood because in the state-space likelihood all demographic parameters are included. Generally, the amount of information about demographic parameters in the state-space likelihood is small. If it is combined with a likelihood that contains plenty of information about



**FIGURE 11.3** Graphical representation of different integrated population models. This graph is similar to an acyclic directed graph (DAG) without the priors. Small squares represent the data, circles the parameters (blue: target parameters, green: nuisance parameters), large squares the individual submodels, and arrows the flux of information. Circles appearing in two submodels indicate that they are informed from two data sources. (a) Model of Section 11.3; (b) same model as (a), but without the availability of productivity data (Section 11.4); (c) model of Section 11.6. For the notation of the parameters and data see text.

a demographic parameter, as is the case with the capture–recapture data and survival, the joint likelihood in terms of survival is dominated by the capture–recapture likelihood. Consequently, nonindependence plays a minor role. In contrast, if the state-space likelihood is combined with a likelihood that contains little information about a demographic parameter, as is the case with the mark-recovery likelihood and survival, the joint likelihood with respect to survival contains similar amounts of information from both the population count and the mark-recovery data. In that case, violation of the independence assumption can have a more serious effect on parameter accuracy. Thus, whether the violation of the independence assumption has an effect on parameter accuracy needs to be evaluated for each model separately.

## 11.3 EXAMPLE OF A SIMPLE IPM (COUNTS, CAPTURE–RECAPTURE, REPRODUCTION)

### 11.3.1 Load Data

We will load one data set from the ortolan bunting population that was simulated using the code in Web appendix 2.

```
# Population counts (from years 1 to 10)
y <- c(45, 48, 44, 59, 62, 62, 55, 51, 46, 42)

# Capture-recapture data (in m-array format, from years 1 to 10)
m <- matrix(c(11, 0, 0, 0, 0, 0, 0, 0, 0, 70,
             0, 12, 0, 1, 0, 0, 0, 0, 0, 52,
             0, 0, 15, 5, 1, 0, 0, 0, 0, 42,
             0, 0, 0, 8, 3, 0, 0, 0, 0, 51,
             0, 0, 0, 0, 4, 3, 0, 0, 0, 61,
             0, 0, 0, 0, 0, 12, 2, 3, 0, 66,
             0, 0, 0, 0, 0, 0, 16, 5, 0, 44,
             0, 0, 0, 0, 0, 0, 0, 12, 0, 46,
             0, 0, 0, 0, 0, 0, 0, 0, 11, 71,
             10, 2, 0, 0, 0, 0, 0, 0, 0, 13,
             0, 7, 0, 1, 0, 0, 0, 0, 0, 27,
             0, 0, 13, 2, 1, 1, 0, 0, 0, 14,
             0, 0, 0, 12, 2, 0, 0, 0, 0, 20,
             0, 0, 0, 0, 10, 2, 0, 0, 0, 21,
             0, 0, 0, 0, 0, 11, 2, 1, 1, 14,
             0, 0, 0, 0, 0, 0, 12, 0, 0, 18,
             0, 0, 0, 0, 0, 0, 0, 11, 1, 21,
             0, 0, 0, 0, 0, 0, 0, 0, 10, 26), ncol = 10,
             byrow = TRUE)
```

The last column in matrix **m** contains the number of released individuals that are never recaptured. The top half of the array contains the data on birds marked as juveniles and the bottom half on those marked as adults.

```
# Productivity data (from years 1 to 9)
J <- c(64, 132, 86, 154, 156, 134, 116, 106, 110)
R <- c(21, 28, 26, 38, 35, 33, 31, 30, 33)
```

Vector **J** contains the total number of nestlings recorded, and vector **R** is the annual number of surveyed broods. The numbers recorded in the last year are not considered here because they are not needed in the population model.

### 11.3.2 Analysis of the Model

We find it useful to highlight three sections in the BUGS code for analyzing the integrated population model. First, we define the priors of the unknowns (parameters, latent effects). This includes the latent population size of each class in the first year, the demographic parameters, and the observation error. As always in this book, we can specify vague priors with little prior information if we wish. We must ensure that the priors have support only for values that are possible at all (e.g., positive numbers for the initial population sizes). In our model, all demographic rates are constant over time; hence, we define priors for their means. When using vague priors for the initial population sizes, BUGS may not even start to update or, if it does, the chains may struggle to converge. Thus, with integrated population models, it is often advisable to use slightly more informative priors. King et al. (2010) suggest normal priors centered on the observed count in the first year and with a variance that is equal to the observation error.

Second, we may compute various derived parameters. As we have seen many times, one of the big assets of an MCMC-based Bayesian analysis is the ease with which derived quantities can be computed along with a full assessment of their uncertainty. In our example, we are interested in the population growth rate.

Third, our code contains likelihoods of the different data sets whose modeling we integrate in our analysis. These likelihoods are more or less identical to the BUGS code given for the different data sets so far, for example, in Chapters 5 and 7. You may ask yourself the following question: where is that ugly joint likelihood that we saw in [Section 11.2.3](#)? Does it lurk in some terrifying BUGS code that we have yet to disclose? No, what you see is all that is necessary to define the joint likelihood of the model in the BUGS language. To us, as ecologists, this is another great asset of Bayesian population analyses using BUGS: instead of defining huge likelihoods at once, you decompose them and describe them by defining the quantities in the model and their stochastic and deterministic local relationships. The joint likelihood is then defined

implicitly by using the same names for some parameters that co-occur in different component likelihoods. We believe that a good grasp of reasonably complex statistical models such as an IPM is much more within the reach of most ecologists than it is when you want to use the method of maximum likelihood to fit the same model. Thus, again, BUGS frees the modeler in you.

```
# Specify model in BUGS language
sink("ipm.bug")
cat("
model {

# -----
# Integrated population model
# - Age structured model with 2 age classes:
#     1-year old and adults (at least 2 years old)
# - Age at first breeding = 1 year
# - Prebreeding census, female-based
# - All vital rates assumed to be constant
# -----


# -----
# 1. Define the priors for the parameters
# -----


# Observation error
tauy <- pow(sigma.y, -2)
sigma.y ~ dunif(0, 50)
sigma2.y <- pow(sigma.y, 2)

# Initial population sizes
N1[1] ~ dnorm(100, 0.0001)I(0,)           # 1-year
Nad[1] ~ dnorm(100, 0.0001)I(0,)          # Adults

# Survival and recapture probabilities, as well as productivity
for (t in 1:(nyears-1)){
    sjuv[t] <- mean.sjuv
    sad[t] <- mean.sad
    p[t] <- mean.p
    f[t] <- mean.fec
}
mean.sjuv ~ dunif(0, 1)
mean.sad ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.fec ~ dunif(0, 20)

# -----
# 2. Derived parameters
# -----


# Population growth rate
for (t in 1:(nyears-1)){
    lambda[t] <- Ntot[t+1] / Ntot[t]
}
```

```

# -----
# 3. The likelihoods of the single data sets
# -----
# 3.1. Likelihood for population population count data (state-space
model)
# 3.1.1 System process
for (t in 2:nyears){
  mean1[t] <- f[t-1] / 2 * sjuv[t-1] * Ntot[t-1]
  N1[t] ~ dpois(mean1[t])
  Nad[t] ~ dbin(sad[t-1], Ntot[t-1])
}
for (t in 1:nyears){
  Ntot[t] <- Nad[t] + N1[t]
}

# 3.1.2 Observation process
for (t in 1:nyears){
  y[t] ~ dnorm(Ntot[t], tauy)
}

# 3.2 Likelihood for capture-recapture data: CJS model (2 age classes)
# Multinomial likelihood
for (t in 1:2*(nyears-1)){
  m[t,1:nyears] ~ dmulti(pr[t,], r[t])
}

# Calculate the number of released individuals
for (t in 1:2*(nyears-1)){
  r[t] <- sum(m[t,])
}

# m-array cell probabilities for juveniles
for (t in 1:(nyears-1)){
  # Main diagonal
  q[t] <- 1-p[t]
  pr[t,t] <- sjuv[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    pr[t,j] <- sjuv[t]*prod(sad[(t+1):j])*prod(q[t:(j-1)])*p[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr[t,j] <- 0
  } #j
  # Last column: probability of non-recapture
  pr[t,nyears] <- 1-sum(pr[t,1:(nyears-1)])
} #t

# m-array cell probabilities for adults
for (t in 1:(nyears-1)){
  # Main diagonal
  pr[t+nyears-1,t] <- sad[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    pr[t,j] <- 0
  } #j
  # Last column: probability of non-recapture
  pr[t,nyears] <- 1-sum(pr[t,1:(nyears-1)])
} #t

```

```

pr[t+nyears-1,j] <- prod(sad[t:j])*prod(q[t:(j-1)])*p[j]
} #j
# Below main diagonal
for (j in 1:(t-1)){
  pr[t+nyears-1,j] <- 0
} #j
# Last column
pr[t+nyears-1,nyears] <- 1 - sum(pr[t+nyears-1,1:(nyears-1)])
} #t

# 3.3. Likelihood for productivity data: Poisson regression
for (t in 1:(nyears-1)){
  J[t] ~ dpois(rho[t])
  rho[t] <- R[t]*f[t]
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(m = m, y = y, J = J, R = R, nyyears = dim(m)[2])

# Initial values
inits <- function(){list(mean.sjuv = runif(1, 0, 1), mean.sad = runif
(1, 0, 1), mean.p = runif(1, 0, 1), mean.fec = runif(1, 0, 10),
N1 = rpois(dim(m)[2], 30), Nad = rpois(dim(m)[2], 30), sigma.y = runif
(1, 0, 10))}

# Parameters monitored
parameters <- c("mean.sjuv", "mean.sad", "mean.p", "mean.fec", "N1",
"Nad", "Ntot", "lambda", "sigma2.y")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 2 min)
ipm <- bugs(bugs.data, inits, parameters, "ipm.bug", n.chains = nc,
n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
bugs.dir, working.directory = getwd())

```

The chains converge fairly quickly. Here are the posterior summaries:

```

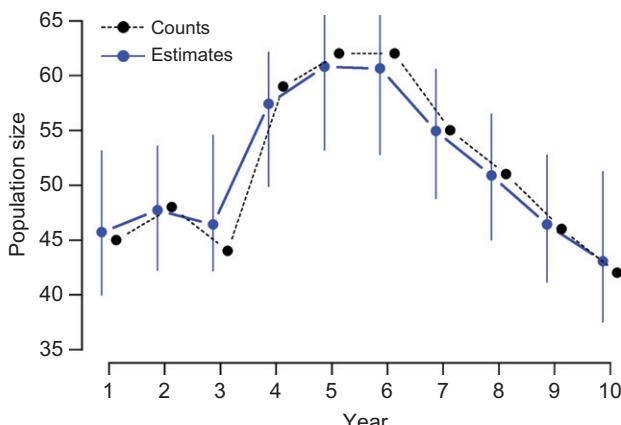
print(ipm, digits = 3)
      mean     sd   2.5%    25%    50%    75%   97.5%   Rhat n.eff
mean.sjuv  0.257  0.018  0.223  0.245  0.257  0.269  0.294  1.001  7500
mean.sad   0.519  0.026  0.468  0.501  0.519  0.536  0.570  1.001  7500
mean.p     0.619  0.037  0.548  0.594  0.618  0.644  0.691  1.001  7500
mean.fec   3.829  0.115  3.603  3.752  3.826  3.907  4.055  1.003  1000
N1[1]      22.757 13.264  1.127 11.440 22.600 34.052 45.310  1.009  320
[ ... ]
N1[10]     20.237  3.563 13.630 17.880 20.050 22.490 27.565  1.001  4900
Nad[1]      22.955 13.209  1.359 11.330 23.050 34.052 45.180  1.005  540
[ ... ]

```

Nad[10]	22.831	3.230	17.000	21.000	23.000	25.000	29.000	1.002	1400
Ntot[1]	45.711	3.101	39.990	44.260	45.220	46.870	53.137	1.001	5100
[ ... ]									
Ntot[10]	43.068	3.240	37.540	41.480	42.400	44.310	51.241	1.001	3000
lambda[1]	1.047	0.071	0.893	1.008	1.055	1.082	1.192	1.001	7500
[ ... ]									
lambda[9]	0.930	0.068	0.804	0.893	0.920	0.960	1.096	1.001	7500
sigma2.y	14.580	28.382	0.016	1.550	6.087	16.480	79.205	1.010	740

Estimated population sizes are quite close to the counts in our example (Fig. 11.4). As expected, the estimates are less variable than the counts, illustrating the smoothing that results from separating out state and observation processes, the autoregressive nature of the population model, and adding more information (from the other data sets). We could also plot the age-specific population sizes, but in our study example this is not very interesting since both age classes have about the same size. This is because the stable age distribution (obtained as the right eigenvector of the Leslie matrix) in this population is 50% 1-year-old individuals and 50% adults. Yet, generally, this is an advantage of integrated models: one may obtain estimates of the size of population segments (such as age classes) that are never even observed.

```
# Produce Fig. 11-4
par(cex = 1.2)
lower <- upper <- numeric()
for (i in 1:10) {
  lower[i] <- quantile(ipm$sims.list$Ntot[,i], 0.025)
  upper[i] <- quantile(ipm$sims.list$Ntot[,i], 0.975)
}
```



**FIGURE 11.4** Observed (black) and estimated population sizes (blue) with 95% CRI under an integrated population model for the simulated ortolan bunting data set.

```

plot(ipm$mean$Ntot, type = "b", ylim = c(35, 65), ylab = "Population size",
     xlab = "Year", las = 1, pch = 16, col = "blue", frame = F, cex = 1.5)
segments(1:10, lower, 1:10, upper, col = "blue")
points(y, type = "b", col = "black", pch = 16, lty = 2, cex = 1.5)
legend(x = 1, y = 65, legend = c("Counts", "Estimates"), pch = c(16, 16),
       col = c("black", "blue"), lty = c(2, 1), bty = "n")

```

## **11.4 ANOTHER EXAMPLE OF AN IPM: ESTIMATING PRODUCTIVITY WITHOUT EXPLICIT PRODUCTIVITY DATA**

One very important advantage of integrated population models is that they typically allow one to estimate demographic parameters for which no explicit data are available (e.g., Besbeas et al., 2002; Schaub et al., 2007). This is possible because the population count data contain information about all demographic parameters, and this information is exploited with the integrated population model. Here, we aim to estimate productivity ( $f$ ) in a study in which no explicit data on productivity are available. We focus on the same ortolan bunting population as in the previous section, but this time only use the population count data and the capture–recapture data (Fig. 11.3b), and we adopt the same model structure as before (e.g., constant demographic rates). Essentially, the BUGS code requires one simple change: just remove the definition of the likelihood of the productivity data.

```

# Specify model in BUGS language
sink("ipm-prod.bug")
cat("
model {

# -----
# Integrated population model
# - Age structured model with 2 age classes:
#   1-year old and adults (at least 2 years old)
# - Age at first breeding = 1 year
# - Prebreeding census, female-based
# - All vital rates assumed to be constant
# -----


# -----
# 1. Define the priors for the parameters
# -----


# Observation error
tauy <- pow(sigma.y, -2)
sigma.y ~ dunif(0, 50)
sigma2.y <- pow(sigma.y, 2)

# Initial population sizes
N1[1] ~ dnorm(100, 0.0001) I(0,)      # 1-year
Nad[1] ~ dnorm(100, 0.0001) I(0,)      # Adults

```

```

# Survival and recapture probabilities, as well as productivity
for (t in 1:(nyears-1)) {
  sjuv[t] <- mean.sjuv
  sad[t] <- mean.sad
  p[t] <- mean.p
  f[t] <- mean.fec
}
mean.sjuv ~ dunif(0, 1)
mean.sad ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.fec ~ dunif(0, 20)

# -----
# 2. Derived parameters
# -----
# Population growth rate
for (t in 1:(nyears-1)){
  lambda[t] <- Ntot[t+1] / Ntot[t]
}

# -----
# 3. The likelihoods of the single data sets
# -----
# 3.1. Likelihood for population population count data (state-space
model)
# 3.1.1 System process
for (t in 2:nyears){
  mean1[t] <- f[t-1] / 2 * sjuv[t-1] * Ntot[t-1]
  N1[t] ~ dpois(mean1[t])
  Nad[t] ~ dbin(sad[t-1], Ntot[t-1])
}
for (t in 1:nyears){
  Ntot[t] <- Nad[t] + N1[t]
}

# 3.1.2 Observation process
for (t in 1:nyears){
  y[t] ~ dnorm(Ntot[t], tauy)
}

# 3.2 Likelihood for capture-recapture data: CJS model (2 age classes)
# Multinomial likelihood
for (t in 1:2*(nyears-1)){
  m[t,1:nyears] ~ dmulti(pr[t,], r[t])
}

# Calculate the number of released individuals
for (t in 1:2*(nyears-1)){
  r[t] <- sum(m[t,])
}

# m-array cell probabilities for juveniles
for (t in 1:(nyears-1)){
  # Main diagonal
  q[t] <- 1-p[t]
}

```

```

pr[t,t] <- sjuv[t] * p[t]
# Above main diagonal
for (j in (t+1):(nyears-1)){
  pr[t,j] <- sjuv[t]*prod(sad[(t+1):j])*prod(q[t:(j-1)])*p[j]
  } #j
# Below main diagonal
for (j in 1:(t-1)){
  pr[t,j] <- 0
  } #j
# Last column: probability of non-recapture
pr[t,nyears] <- 1-sum(pr[t,1:(nyears-1)])
} #t

# m-array cell probabilities for adults
for (t in 1:(nyears-1)){
  # Main diagonal
  pr[t+nyears-1,t] <- sad[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    pr[t+nyears-1,j] <- prod(sad[t:j])*prod(q[t:(j-1)])*p[j]
    } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr[t+nyears-1,j] <- 0
    } #j
  # Last column
  pr[t+nyears-1,nyears] <- 1 - sum(pr[t+nyears-1,1:(nyears-1)])
  } #t
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(m = m, y = y, nyyears = dim(m) [2])

# Initial values
inits<- function(){list(mean.sjuv = runif(1, 0, 1), mean.sad = runif(1,
  0, 1), mean.p = runif(1, 0, 1), mean.fec = runif(1, 0, 10), N1 = rpois
  (dim(m) [2], 30), Nad = rpois(dim(m) [2], 30), sigma.y = runif(1, 0,
  10))}

# Parameters monitored
parameters <- c("mean.sjuv", "mean.sad", "mean.p", "mean.fec", "N1",
  "Nad", "Ntot", "lambda", "sigma2.y")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
ipm.prod <- bugs(bugs.data, inits, parameters, "ipm-prod.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

```
# Summarize posteriors
print(ipm.prod, digits = 3)
      mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
mean.sjuv 0.266 0.024 0.222 0.250 0.265 0.282 0.317 1.001 7500
mean.sad  0.527 0.029 0.471 0.507 0.527 0.546 0.584 1.002 1900
mean.p    0.611 0.040 0.531 0.584 0.610 0.639 0.689 1.001 3500
mean.fec  3.547 0.487 2.673 3.209 3.515 3.858 4.557 1.001 5400
N1[1]    23.363 13.252 1.184 12.557 23.275 34.402 45.535 1.014 170
[ ... ]
N1[10]   19.499 3.631 12.699 17.090 19.400 21.750 26.870 1.004 680
Nad[1]   22.561 13.274 1.233 11.390 22.590 33.230 45.305 1.022 120
[ ... ]
Nad[10]  23.398 3.313 17.000 21.000 23.000 26.000 30.000 1.002 1500
Ntot[1]  45.925 3.244 40.600 44.450 45.260 46.960 53.905 1.001 7500
[ ... ]
Ntot[10] 42.898 3.130 37.519 41.470 42.290 43.940 50.430 1.002 1600
lambda[1] 1.044 0.069 0.891 1.007 1.054 1.078 1.187 1.001 7500
[ ... ]
lambda[9] 0.928 0.066 0.804 0.894 0.917 0.957 1.091 1.001 7100
sigma2.y 14.000 31.952 0.003 1.268 5.318 15.167 78.404 1.025 210
```

Most parameter estimates are nearly identical in both models. Overall, the precision of the estimates is slightly reduced now since we have ignored part of the available information by not using the productivity data. The most striking difference occurs for the productivity parameter. Under the model without productivity data, the estimate is slightly lower (3.547 vs. 3.829) and much less precise ( $sd = 0.487$  vs. 0.115). A simulation study shows that the productivity estimator from both models is unbiased, but if no productivity data are available the precision is understandably lower (Abadi et al., 2010a).

## 11.5 IPMs FOR POPULATION VIABILITY ANALYSIS

One particularly useful feature of integrated population models is that predictions of the size of “unobserved populations” (e.g., age classes) can be made. We have already mentioned that we may estimate the size of unobserved segments of a population. However, we may just as well estimate the size of (as yet) unobserved populations in the future. In particular, Bayesian population analysis using posterior sampling is strikingly useful for a full assessment of all uncertainties involved in such forecasts. To project the model into the future and forecast population size, we extend the loops of the state process and adapt the prior definitions of the demographic rates to cover the additional years. Within the MCMC samples, the future population sizes are estimated

with full accounting for all uncertainty in the parameter estimates. The posterior distributions of the predicted future population sizes can be directly used to compute population extinction probabilities or population prediction intervals, both typical inferential targets in population viability analyses (Beissinger, 2002). We illustrate predictions with the ortolan bunting example.

```
# Specify model in BUGS language
sink("ipm-pred.bug")
cat("
model {
# -----
# Integrated population model
# - Age structured model with 2 age classes:
#   1-year old and adults (at least 2 years old)
# - Age at first breeding = 1 year
# - Prebreeding census, female-based
# - All vital rates assumed to be constant
#
# -----
#
# 1. Define the priors for the parameters
#
# -----
# Observation error
tauy <- pow(sigma.y, -2)
sigma.y ~ dunif(0, 50)
sigma2.y <- pow(sigma.y, 2)

# Initial population sizes
N1[1] ~ dnorm(100, 0.0001)I(0,)           # 1-year
Nad[1] ~ dnorm(100, 0.0001)I(0,)           # Adults

# Survival and recapture probabilities, as well as productivity
for (t in 1:(nyears-1+t.pred)){
  sjuv[t] <- mean.sjuv
  sad[t] <- mean.sad
  p[t] <- mean.p
  f[t] <- mean.fec
}

mean.sjuv ~ dunif(0, 1)
mean.sad ~ dunif(0, 1)
mean.p ~ dunif(0, 1)
mean.fec ~ dunif(0, 20)

#
# 2. Derived parameters
#
# -----
# Population growth rate
for (t in 1:(nyears-1+t.pred)){
  lambda[t] <- Ntot[t+1] / Ntot[t]
}
```

```

# -----
# 3. The likelihoods of the single data sets
# -----
# 3.1. Likelihood for population population count data (state-space
model)
# 3.1.1 System process
for (t in 2:nyears+1.pred) {
  mean1[t] <- f[t-1] / 2 * sjuv[t-1] * Ntot[t-1]
  N1[t] ~ dpois(mean1[t])
  Nad[t] ~ dbin(sad[t-1], Ntot[t-1])
}
for (t in 1:nyears+1.pred) {
  Ntot[t] <- Nad[t] + N1[t]
}

# 3.1.2 Observation process
for (t in 1:nyears) {
  y[t] ~ dnorm(Ntot[t], tauy)
}

# 3.2 Likelihood for capture-recapture data: CJS model (2 age classes)
# Multinomial likelihood
for (t in 1:2*(nyears-1)) {
  m[t,1:nyears] ~ dmulti(pr[t,,], r[t])
}

# Calculate the number of released individuals
for (t in 1:2*(nyears-1)) {
  r[t] <- sum(m[t,])
}

# m-array cell probabilities for juveniles
for (t in 1:(nyears-1)) {
  # Main diagonal
  q[t] <- 1-p[t]
  pr[t,t] <- sjuv[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)) {
    pr[t,j] <- sjuv[t]*prod(sad[(t+1):j]) *prod(q[t:(j-1)]) *p[j]
  } #j
  # Below main diagonal
  for (j in 1:(t-1)) {
    pr[t,j] <- 0
  } #j
  # Last column: probability of non-recapture
  pr[t,nyears] <- 1-sum(pr[t,1:(nyears-1)])
} #t

# m-array cell probabilities for adults
for (t in 1:(nyears-1)) {
  # Main diagonal
  pr[t+nyears-1,t] <- sad[t] * p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)) {
    pr[t+nyears-1,j] <- prod(sad[t:j]) *prod(q[t:(j-1)]) *p[j]
  } #j
}

```

```

# Below main diagonal
for (j in 1:(t-1)){
  pr[t+nyears-1,j] <- 0
} #j
# Last column
pr[t+nyears-1,nyears] <- 1 - sum(pr[t+nyears-1,1:(nyears-1)])
} #t

# 3.3. Likelihood for productivity data: Poisson regression
for (t in 1:(nyears-1)){
  J[t] ~ dpois(rho[t])
  rho[t] <- R[t]*f[t]
}
}

",fill = TRUE)
sink()

# Give the number of future years for which population size shall be estimated
t.pred <- 5

# Bundle data
bugs.data <- list(m = m, y = y, J = J, R = R, nyears = dim(m) [2],
  t.pred = t.pred)

# Initial values
inits <- function(){list(mean.sjuv = runif(1, 0, 1), mean.sad = runif
  (1, 0, 1), mean.p = runif(1, 0, 1), mean.fec = runif(1, 0, 10), N1 =
  rpois(dim(m) [2]+t.pred, 30), Nad = rpois(dim(m) [2]+t.pred, 30),
  sigma.y = runif(1, 0, 10))}

# Parameters monitored
parameters <- c("mean.sjuv", "mean.sad", "mean.p", "mean.fec", "N1",
  "Nad", "Ntot", "lambda", "sigma2.y")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 5000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
ipm.pred <- bugs(bugs.data, inits, parameters, "ipm-pred.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

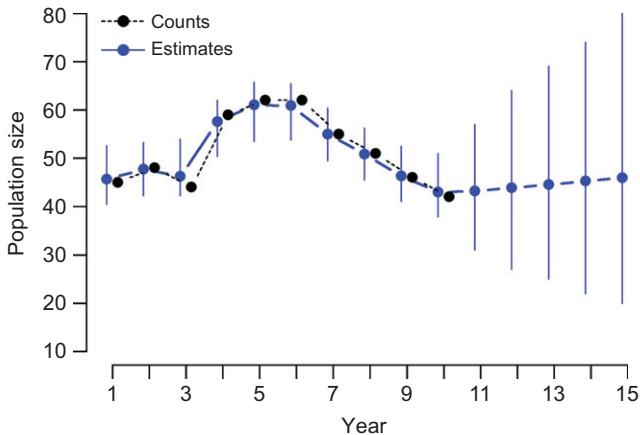
```

The estimated population sizes for the entire 15 years are shown in Fig. 11.5. We notice that 95% CRIs increase over time after year 10, reflecting an increased uncertainty when projecting further ahead. This makes intuitive sense.

```

# Produce Fig. 11-5
par(cex = 1.2)
lower <- upper <- numeric()
for (i in 1:15) {

```



**FIGURE 11.5** Observed (black) and estimated population sizes (blue) along with the 95% CRI. For years 11–15, projected population is shown.

```

lower[i] <- quantile(ipm.pred$sims.list$Ntot[,i], 0.025)
upper[i] <- quantile(ipm.pred$sims.list$Ntot[,i], 0.975)
}
plot(ipm.pred$mean$Ntot, type = "b", ylim = c(10, max(upper)), ylab =
  "Population size", xlab = "Year", las = 1, pch = 16, col = "blue",
  frame = F)
segments(1:15, lower, 1:15, upper, col = "blue")
points(y, type = "b", col = "black", lty = 2, pch = 16)
legend(x = 1, y = 80, legend = c("Counts", "Estimates"), pch = c(16, 16),
  col = c("black", "blue"), lty = c(2, 1), bty = "n")

```

The estimation of extinction probabilities is straightforward. In the example, we might be interested to know the probability that population size falls below 30 pairs in 5 years from now. Our model provides the answer to that based on the assumption that the demographic processes modeled remain the same as observed in the past. We only need to evaluate the posterior distribution of population size in year 15:

```

mean(ipm.pred$sims.list$Ntot[,15]<30)
[1] 0.140

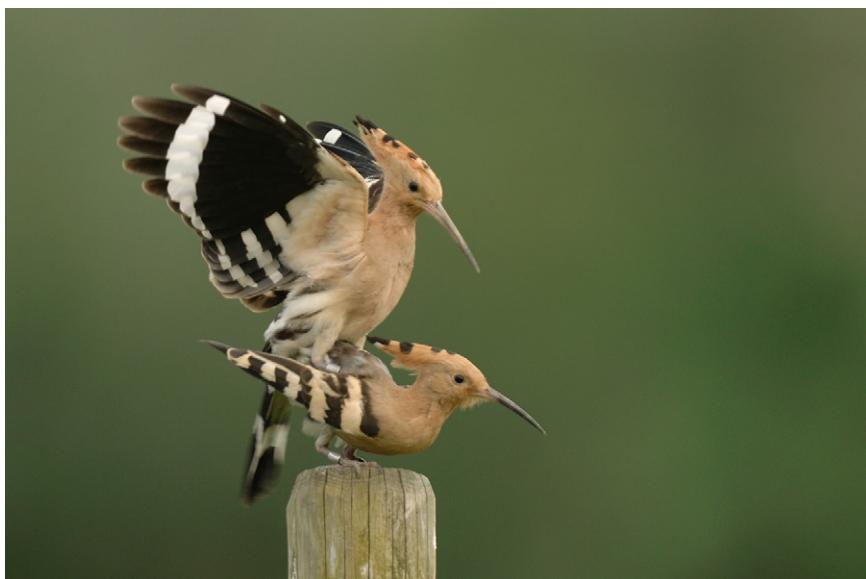
```

Forecasting population size is slightly more complicated if the demographic parameters are not constant but vary over time. The best way to achieve this goal is to estimate a mean and a temporal variance of the demographic parameters from the observed data (see Section 11.6), that is, fit random year effects. Future realizations of the demographic rates can then be sampled from these distributions. When specifying random year effects in WinBUGS, process variability is automatically accounted for.

## 11.6 REAL DATA EXAMPLE: HOOPOE POPULATION DYNAMICS

We look at a real example using data types that are typically collected in demographic population studies. We studied a local population of hoopoes (Fig. 11.6) in the SW Swiss Alps (Valais; Arlettaz et al. 2010). In an area of 62 km<sup>2</sup>, the vast majority of hoopoes use nest boxes for reproduction. From 2002 to 2009, these nest boxes were repeatedly checked every year to record the breeding success, and all the nestlings were ringed. In addition, adults were marked and recaptured. Thus, we obtained the following three types of demographic data: (1) a population count (the maximal number of simultaneously occupied nest boxes in each year), (2) capture–recapture data of adults and young, and (3) data on reproductive output. The goal of our study was to estimate the demographic parameters as well as their temporal variability to understand the dynamics of the hoopoe population (Schaub et al., 2011).

Although the study area is fairly large, the population is not geographically closed. Therefore, emigration and immigration need to be considered. We have already seen in Chapter 7 that capture–recapture data allow estimation of apparent survival probabilities, that is, the product of true survival and site fidelity. By modeling apparent survival in a CJS model as a part of



**FIGURE 11.6** Happy hoopoes (*Upupa epops*), Switzerland (Photograph by P. Keusch). Note that both individuals are ringed; they belong to the study population in the Valais (Schaub et al., 2011).

the integrated population model, we automatically account for emigration, even if we cannot estimate it. On the other hand, information on immigration is very elusive. In our study, the population counts contain information about immigration because the annual change in population size is a result of all four demographic processes operating in a population: survival, productivity, immigration, and emigration. We have independent data for the other demographic processes (capture–recapture for apparent survival, reproductive output for productivity); hence, the combined analysis should enable us to obtain an estimate of immigration. This neat idea was brought up by Abadi et al. (2010b).

To write the BUGS code for our model, we take the three steps outlined earlier. First, we describe the link between population change and demographic rates. Hoopoes are short-lived, and we therefore assume two age classes only. We survey the population just before reproduction; that is, we conduct a prebreeding census. Hence, we differentiate 1-year-old individuals from older individuals. We include demographic stochasticity by using appropriate distributions. Since we want to estimate immigration, the population can be divided into three components: surviving philopatric adults ( $N_{\text{ad}}$ ), locally produced young that survived and recruited into our population ( $N_1$ ), and immigrants ( $N_{\text{im}}$ ). We model the numbers of individuals in these population segments with binomial and Poisson distributions:

$$N_{\text{ad},t+1} \sim \text{Binomial}(N_{1,t} + N_{\text{ad},t} + N_{\text{im},t}, \phi_{\text{ad},t})$$

$$N_{1,t+1} \sim \text{Poisson}\left((N_{1,t} + N_{\text{ad},t} + N_{\text{im},t})\phi_{\text{juv},t} \frac{f_t}{2}\right)$$

$$N_{\text{im},t+1} \sim \text{Poisson}\left((N_{1,t} + N_{\text{ad},t} + N_{\text{im},t})\omega_t\right).$$

Here,  $\phi_{\text{juv},t}$  and  $\phi_{\text{ad},t}$  are apparent survival probabilities from year  $t$  to  $t+1$  of young (from fledging to age 1) and adults, respectively,  $f_t$  is productivity in year  $t$ , and  $\omega_t$  is the immigration rate. The latter is expressed as a ratio, that is, as the number of immigrants in year  $t+1$  per individual in year  $t$ . We could estimate the annual number of immigrants using a Poisson distribution, but we prefer the rate, because we are interested in the temporal variation of the demographic rates and in the temporal variation of components of population size.

The second step is to write the likelihood of the individual data sets. As before, we use a state-space model for the population counts, with the state-process equations given above. The assumed observation process is slightly different from the earlier one: we now adopt a Poisson observation error (i.e.,  $y_t \sim \text{Poisson}(N_{1,t} + N_{\text{ad},t} + N_{\text{im},t})$ ). Hence, the relative observation error is constant, but the absolute observation error increases with population size, because in the Poisson distribution the variance is equal to the mean. Compared to the log-normal distribution used by Schaub

et al. (2011) for the same data, convergence of Markov chains is obtained faster with a Poisson observation error. Parameter estimates are almost identical regardless of the distribution chosen for the observation error (you can try this out for the lognormal or even the normal). The state-space model further requires prior distributions for the initial sizes of all population components ( $N_{1,1}$ ,  $N_{\text{ad},1}$ ,  $N_{\text{im},1}$ ). For the capture–recapture data, we use the CJS model with a multinomial likelihood and for the productivity data a Poisson regression. We assume independence among the three data sets, and consequently, the joint likelihood, the definition of which is our third step, is the product of the three individual data likelihoods. This model is depicted graphically in Fig. 11.3c.

We are particularly interested in assessing the temporal variability of all demographic parameters, and we formulate a model that includes environmental stochasticity. We specify random time effects for all demographic parameters as discussed in Section 7.4.2. Thus, we imagine that the annual values of all demographic rates (on an appropriately transformed scale) are realizations from normal distributions with means and variances that we estimate. We use the logit link for survival probabilities and the log link for productivity and immigration rates. When adopting a hierarchical model for the demographic parameters in each year, we no longer need priors and initial values for each of them in each year. Rather, we specify a prior and an initial value for their mean and variance, that is, for the hyperparameters. We use vague normal priors for the logit of the mean survival probabilities and the log of mean fecundity and immigration and uniform priors for the variance parameters on the standard deviation scale.

Now, we are ready! We load the data, write the model in BUGS language, and run it.



```

log(f[t]) <- l.mfec + epsilon.fec[t]           # Productivity
log(omega[t]) <- l.mim + epsilon.im[t]         # Immigration
logit(p[t]) <- l.p                            # Recapture probability
}

# -----
# 3. Derived parameters
# -----
mphij <- exp(l.mphij)/(1+exp(l.mphij))        # Mean juvenile survival
                                                 probability
mphia <- exp(l.mphia)/(1+exp(l.mphia))        # Mean adult survival
                                                 probability
mfec <- exp(l.mfec)                           # Mean productivity
mim <- exp(l.mim)                            # Mean immigration rate

# Population growth rate
for (t in 1:(nyears-1)){
  lambda[t] <- Ntot[t+1] / Ntot[t]
  logla[t] <- log(lambda[t])
}
mlam <- exp((1/(nyears-1))*sum(logla[1:(nyears-1)]))  # Geometric
                                                               mean

# -----
# 4. The likelihoods of the single data sets
# -----
# 4.1. Likelihood for population population count data (state-space
model)
# 4.1.1 System process
for (t in 2:nyears){
  mean1[t] <- 0.5 * f[t-1] * phij[t-1] * Ntot[t-1]
  N1[t] ~ dpois(mean1[t])
  NadSurv[t] ~ dbin(phia[t-1], Ntot[t-1])
  mpo[t] <- Ntot[t-1] * omega[t-1]
  Nadimm[t] ~ dpois(mpo[t])
}

# 4.1.2 Observation process
for (t in 1:nyears){
  Ntot[t] <- NadSurv[t] + Nadimm[t] + N1[t]
  y[t] ~ dpois(Ntot[t])
}

# 4.2 Likelihood for capture-recapture data: CJS model (2 age classes)
# Multinomial likelihood
for (t in 1:(nyears-1)){
  marray.j[t,1:nyears] ~ dmulti(pr.j[t,], r.j[t])
  marray.a[t,1:nyears] ~ dmulti(pr.a[t,], r.a[t])
}

# Calculate number of released individuals
for (t in 1:(nyears-1)){
  r.j[t] <- sum(marray.j[t,])
  r.a[t] <- sum(marray.a[t,])
}

```

```

# m-array cell probabilities for juveniles
for (t in 1:(nyears-1)){
  q[t] <- 1-p[t]
  # Main diagonal
  pr.j[t,t] <- phij[t]*p[t]
  # Above main diagonal
  for (j in (t+1):(nyears-1)){
    pr.j[t,j] <- phij[t]*prod(phia[(t+1):j])*prod(q[t:(j-1)])*p[j]
    } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr.j[t,j] <- 0
    } #j
  # Last column
  pr.j[t,nyears] <- 1-sum(pr.j[t,1:(nyears-1)])
  } #t

# m-array cell probabilities for adults
for (t in 1:(nyears-1)){
  # Main diagonal
  pr.a[t,t] <- phia[t]*p[t]
  # above main diagonal
  for (j in (t+1):(nyears-1)){
    pr.a[t,j] <- prod(phia[t:j])*prod(q[t:(j-1)])*p[j]
    } #j
  # Below main diagonal
  for (j in 1:(t-1)){
    pr.a[t,j] <- 0
    } #j
  # Last column
  pr.a[t,nyears] <- 1-sum(pr.a[t,1:(nyears-1)])
  } #t

# 4.3. Likelihood for productivity data: Poisson regression
for (t in 1:(nyears-1)){
  J[t] ~ dpois(rho[t])
  rho[t] <- R[t] * f[t]
}
",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(nyears = nyyears, marray.j = marray.j, marray.a =
  marray.a, y = popcount, J = J, R = R)

# Initial values
inits <- function(){list(l.mphij = rnorm(1, 0.2, 0.5), l.mphia = rnorm
  (1, 0.2, 0.5), l.mfec = rnorm(1, 0.2, 0.5), l.mim = rnorm(1, 0.2,
  0.5), l.p = rnorm(1, 0.2, 1), sig.phij = runif(1, 0.1, 10), sig.phia =
  runif(1, 0.1, 10), sig.fec = runif(1, 0.1, 10), sig.im = runif(1, 0.1,
  10), N1 = round(runif(nyears, 1, 50), 0), NadSurv = round(runif(
  nyyears, 5, 50), 0), Nadimm = round(runif(nyears, 1, 50), 0))}
```

```

# Parameters monitored
parameters <- c("phij", "phia", "f", "omega", "p", "lambda", "mphiij",
  "mphia", "mfec", "mim", "mlam", "sig.phij", "sig.phia", "sig.fec",
  "sig.im", "N1", "NadSurv", "Nadimm", "Ntot")

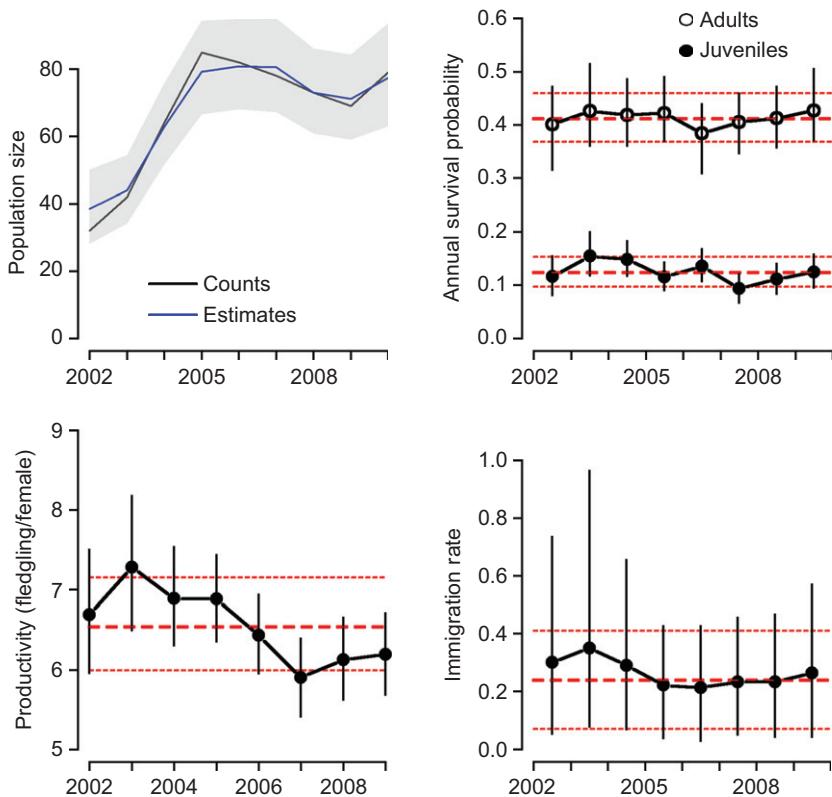
# MCMC settings
ni <- 20000
nt <- 6
nb <- 10000
nc <- 3

# Call WinBUGS from R (BRT 5 min)
ipm.hoopoe <- bugs(bugs.data, inits, parameters, "ipm.hoopoe.bug",
  n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

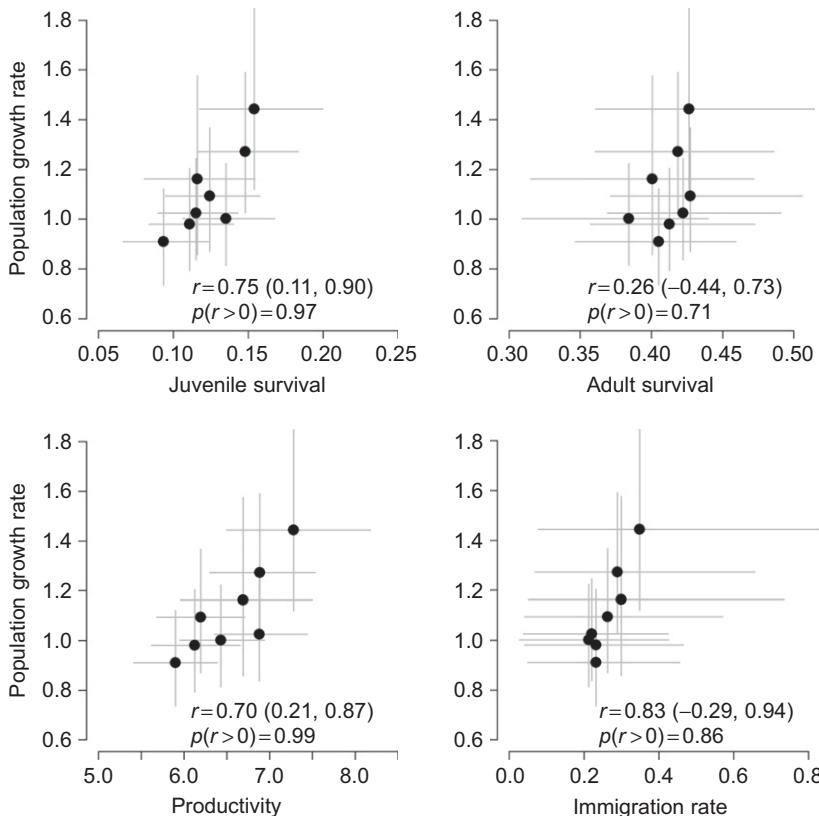
With the chosen MCMC settings, we achieve convergence, though for a research paper, we may decide to run longer chains. Next, we inspect posterior distributions of the parameters of interest. Figure 11.7 is a graphical summary of what we learn about the dynamics of our hoopoe population (see Web appendix 1 for R code). The estimated population size is close to the population counts; that is, the observation error appears small. This is not surprising because the vast majority of hoopoes in our study use nest boxes, and the count of occupied nest boxes is unaffected by the observation error. Yet, there is still some uncertainty because some hoopoes conduct two broods in a year, and first and second broods may overlap. By modeling the maximal number of simultaneously occupied nest boxes, we strictly model an index of the actual population size, though probably a rather accurate one. The uncertainty around the population size estimates is quite large. This may reflect uncertainty due to the estimation of the demographic parameters and especially the uncertainty introduced by accounting for annual variability in all rates. Survival and productivity are estimated quite precisely; that is, they have a narrow CRI. In contrast, the immigration rate is not estimated very precisely because we have no explicit data on immigration (compare this with the similar situation in Section 11.4, in which we estimated productivity without productivity data). Immigration rate is one of the hardest demographic parameters to estimate in a demographic population analysis.

Specification of annual rates as random effects allowed us to estimate the temporal variance in the demographic rates and to get better estimates of the annual demographic rates (Gelman and Hill, 2007). The annual parameter estimates are shrunk toward the mean of their prior distribution, and the degree of shrinkage depends on the precision and the temporal variance (Burnham and White, 2002). Shrinkage pulls back outlying estimates toward more “sensible” values (see Section 4.1).



**FIGURE 11.7** Posterior means (with 95% CRI) of the demographic parameters in a hoopoe population in SW Switzerland under a population model with random year effects for all demographic rates. Red lines show the mean and the 95% CRI of the mean hyperparameters (see Web appendix 1 for R code to create this figure).

To study the link between demographic rates and population growth, we can correlate the estimates of annual population growth rates with those of each demographic rate. These correlations provide indications about the strength of the contributions of the temporal variation in demographic rates to the temporal variation in the population growth rate (Robinson et al., 2004; Schaub et al., 2011). These correlations were positive for all demographic rates (Fig. 11.8; for R code see Web appendix 1). The correlation was highest for productivity and lowest for adult survival, suggesting that variation in productivity contributed more to the variation in population growth rates than the variation in adult survival. Credible intervals for correlation coefficients and the incidence of a positive correlation can easily be computed as derived quantity from the MCMC output of the analysis (see Web appendix 1).



**FIGURE 11.8** Estimates of annual demographic rates plotted against the estimates of interannual population growth. Black dots show posterior means and grey lines 95% CRI. Inset we print the posterior mode of the correlation coefficients ( $r$ , with 95% CRI) and the probability of a positive correlation ( $p(r>0)$ ) (see Web appendix 1 for R code to create this figure).

## 11.7 SUMMARY AND OUTLOOK

In this chapter, we have introduced integrated population models. In this recently developed modeling framework, population counts are combined coherently with data on demographic rates, such as survival or productivity, in a sort of stochastic Leslie matrix model. There are many advantages of using an integrated population model: the precision of parameter estimates is increased and parameters that are not estimable otherwise may become estimable. Thus, an integrated population model is much more powerful than many traditional approaches to population analysis, in which counts and data on demographic rates are analyzed

separately. The core of any integrated model is a population projection matrix model, which provides the explicit link between changes in population size and demographic rates. A state-space formulation for the counts separates the dynamics of the true population size from observation error.

Using integrated population models allows one to make rigorous inference about unobserved quantities, such as the sizes of all age (or stage) classes defined, even if we cannot distinguish them in the field. As one caveat, though, we must not forget that what is called “population size” in state-space models such as those in this chapter and in Chapter 5 is really a smoothed index of population size; see Chapter 5. If our goal is to get unbiased estimates of population size, additional information about the population count data is necessary, combined with a different model for the observation process (i.e., as in Chapters 6, 10, 12, and 13).

So far no goodness-of-fit test is available for integrated population models (Schaub and Abadi, 2011). The best that can be done at the moment is to assess the goodness of fit of individual likelihoods to single data set using established procedures. If these tests are satisfactory, we may assume that the integrated model is also a satisfactory description of the complete data.

Integrated population models are a powerful and flexible framework to understand population dynamics (Schaub and Abadi, 2011). Our examples combined rather simple Leslie matrix state-space, capture–recapture, and Poisson regression models. Yet, it is easy to either make these models more complicated and realistic, consider multistate models or likelihoods of different data types. For example, Brooks et al. (2004) used mark-recovery data of lapwings instead of capture–recapture data to get independent information about true rather than apparent survival (see Exercise 4 in Section 11.8). Schaub et al. (2010) used age-at-death data to inform survival in an eagle owl population. The basic structure of the integrated model remains the same; you just need to replace the different modules in the joint likelihood, and you can make creative use of all the Lego pieces that we have described in this book. Thus, the models featured in this chapter provide some sort of synthesis of all other models, which makes them extremely flexible and powerful!

## 11.8 EXERCISES

1. Predict the hoopoe population size in 3 years. How large is the extinction probability (assume an extinction threshold of five pairs)?
2. Assume that population count data from years 3 and 5 are missing in the hoopoe example. Use an integrated population model to estimate these missing data. What do you observe?

3. Fit an integrated population model with time-dependent parameters to the ortolan bunting data (Section 11.3). Compare the population size estimates with those from a model with constant demographic parameters and explain.
4. Use an integrated population model to study population dynamics of British lapwings (Besbeas et al., 2002; Brooks et al., 2004). The data consist of a national population index (1965–1998) and of mark recoveries from individuals marked as hatchings (1963–1997). No data on productivity is available. Construct a model with two age classes for survival and where (1) first breeding occurs at the age of 2 years and (2) where it occurs at the age of 3 years. Further, (3) make a model where survival is a function of the number of frost days. The data file `lapwing.txt` (population index, m-array, normalized number of frost days; from Brooks et al. 2004) can be found on the book website ([www.vogelwarte.ch/bpa](http://www.vogelwarte.ch/bpa)).

# 12

## Estimation of Abundance from Counts in Metapopulation Designs Using the Binomial Mixture Model

### OUTLINE

12.1 Introduction	383
12.2 Generation and Analysis of Simulated Data	388
12.2.1 The Simplest Case with Constant Parameters	388
12.2.2 Introducing Covariates	390
12.3 Analysis of Real Data: Open-Population Binomial Mixture Models	396
12.3.1 Simple Poisson Model	398
12.3.2 Zero-Inflated Poisson Binomial Mixture Model (ZIP Binomial Mixture Model)	401
12.3.3 Binomial Mixture Model with Overdispersion in Both Abundance and Detection	404
12.4 Summary and Outlook	409
12.5 Exercises	411

### 12.1 INTRODUCTION

The last two chapters of this book deal with the modeling of abundance and occurrence in a metapopulation design. We show how a sort of non-standard generalized linear mixed model (GLMM), a logistic regression

with Poisson or with Bernoulli random effects, can be used to estimate population size or species occurrence in systems of spatial replicate populations. At least some of them must be surveyed more than once during a short period, that is, replication is required in space *and* in time. Short means that the dynamics of the collection of populations (extinction, colonization, or emigration and immigration, as well as survival and recruitment) must be negligible over the time period over which replicate surveys are conducted. We call the design of studies with such systems of spatial replicate populations with temporally replicated samples a *metapopulation design* (Royle, 2004c; Kéry and Royle, 2010). The system of spatial replicates may or may not be inhabited by a metapopulation in the biological sense (Hanski, 1994, 1998).

In Chapters 4 and 5, we used two variants of hierarchical model that attempt to partition the observed data into contributions from the dynamics of the true ecological state and from an observation process. These models are useful because failure to distinguish between the two processes that generate the observed data will often lead to severely biased inferences, for example, about the presence or magnitude of density dependence. However, if any further information is absent (e.g., covariates informative about the observation process), these models cannot account for patterns in detection probability (see Link and Sauer, 1998 for good examples of such covariate modeling as a partial remedy for imperfect and patterned detection probability). The models in Chapters 4 and 5 are unable in principle to fully correct for the observation error. We have shown that with a single count per time interval, this framework simply models the expectation  $Np$ , where  $N$  is population size, and  $p$  is the average detection probability. What in these models is called “observation error” is simply the ups and downs around  $Np$  of the observed counts due to binomial sampling error. Therefore, such hierarchical models have been called implicit hierarchical models (Royle and Dorazio, 2008): the product  $Np$  is not a quantity with an explicit biological meaning.

In the present chapter, we extend these implicit hierarchical models for counts to become explicit hierarchical models for counts so that the two main parameters have the interpretation of local abundance  $N$  and of detection probability  $p$ . The binomial (also called N-) mixture model of Royle (2004c) jointly estimates local abundance and detection probability (Dodd and Dorazio, 2004; Royle, 2004a, 2004c; Kéry et al., 2005b; Dorazio, 2007; Royle and Dorazio, 2008; Wenger and Freeman, 2008; Joseph et al., 2009; Kéry et al., 2009a; Kéry and Royle, 2010; Post van der Burg et al., 2011). It takes as input spatially and temporally replicated counts of independent individuals within a period of closure and yields estimates of the parameters of the ecological and the observation processes. The parameters of the ecological process describe the spatial or spatio-temporal

variation in latent abundance. Abundance is a latent state because it is incompletely observed owing to detection error. The observation process is the process of detecting (or overlooking) individuals and is described by a binomial process as we have seen so many times.

The binomial mixture model has great appeal, since it allows us to estimate abundance, corrected for imperfect detection, from fairly “cheap” data: simple counts without any extra information, such as individual identification or distance measurements. Therefore, this model may be more widely applicable than capture–recapture methods or distance sampling. What the model does require, though, is replication in two dimensions, both spatially (at >10–20 sites, say) and temporally within a period of closure (at least two observations per site, though not necessarily at every site). Hence, it requires counts  $y_{i,j}$  for a number of sites  $i$  and temporal replicates  $j$ .

Conceptually, such replicated counts arise from two distinct processes, one ecological and another observational. Accordingly, the simplest binomial mixture model for a single period of closure (season) can be written succinctly in just two lines:

$$\begin{aligned} N_i &\sim \text{Poisson}(\lambda) & 1. \text{ Ecological process yields latent state} \\ y_{i,j} | N_i &\sim \text{Binomial}(N_i, p) & 2. \text{ Observation process yields observations} \end{aligned}$$

First, the spatial variation of local abundance at site  $i$ ,  $N_i$ , for a collection of sites is described by a Poisson distribution with mean  $\lambda$ . Second, the observed counts  $y_{i,j}$  (given  $N_i$ ) at site  $i$  and during replicate survey  $j$  are described by a binomial distribution with sample size  $N_i$  and detection probability  $p$ . This model is sometimes also called a Poisson-binomial mixture model.

If we think of it, few things could be more natural than making these two distributional assumptions. The Poisson is the standard distribution assumed for spatial or temporal variation in abundance (McCullagh and Nelder, 1989), and the binomial distribution underlies a vast array of capture–recapture models as a formal description of the coin-flip-like detection process of individuals (Williams et al., 2002; see also Section 1.3). Thus, the binomial mixture model can be called a hierarchical Poisson regression, since the basic model for abundance is Poisson, but there is a logistic regression attached to account for imperfect detection. Several important extensions are possible, among them the adoption of distributions other than the Poisson for abundance or the binomial for detection, the introduction of covariates, and the relaxation of the independence of detection and of the closure assumption. We next briefly sketch each one in turn.

First, we could specify distributions other than a Poisson for the ecological part of the model, for instance, a negative binomial (Royle, 2004b, 2004c; Kéry et al., 2005b; Joseph et al., 2009), or a zero-inflated Poisson (Wenger and Freeman, 2008; Joseph et al., 2009). We will see a zero-inflated Poisson in [Section 12.3.2](#) and the Poisson log-normal as an alternative to a negative binomial distribution in [Section 12.3.3](#).

Second, since the binomial mixture model consists simply of two linked GLMs, it is natural to introduce the effects of covariates through a log- and a logit-link function, respectively, for abundance and detection. This means that neither the mean local abundance  $\lambda$  nor detection probability  $p$  needs to be constant; indeed, they rarely ever are! Rather, we can specify covariate relationships such as these:

$$\log(\lambda_i) = \alpha_0 + \alpha_1 * x_i \text{ and}$$

$$\text{logit}(p_{i,j}) = \beta_0 + \beta_1 * x_{i,j}$$

In the first case, mean abundance at site  $i$ , on the scale of the natural logarithm, is a linear function of site-specific covariate  $x_i$  (a “site covariate”), with intercept  $\alpha_0$  and slope  $\alpha_1$ . Thus, the binomial mixture model allows one to model habitat relationships in abundance while accounting for detection error; for examples, see Kéry (2008), Webster et al. (2008), Chandler et al. (2009a, b), and Schlossberg et al. (2010). In the second case, the logit transform of detection probability at site  $i$  during survey  $j$  is a logit-linear function of the site- and survey-specific covariate  $x_{i,j}$  (a “survey covariate”), with intercept  $\beta_0$  and slope  $\beta_1$ . Of course, a site covariate ( $x_i$ ) is also possible for detection.

Similarly, for abundance or detection, latent structure can be added by the introduction of random effects on the scale of the linear predictor. These account for additional variation in abundance or detection that is not accounted for by the nominal distributional assumptions along with the specified covariates. For instance, we can model extra-Poisson dispersion in the latent abundance parameters  $N_i$  by specifying the following Poisson-log-normal binomial mixture model (only linear predictor shown):

$$\log(\lambda_i) = \alpha_0 + \alpha_1 * x_i + \varepsilon_i,$$

$$\text{with } \varepsilon_i \sim \text{Normal}(0, \sigma_\lambda^2)$$

This modeling can be seen as a sort of correction for overdispersion in abundance (see also [Section 4.2](#)). Its result will be similar to the adoption of a negative binomial distribution instead of a Poisson. Random effects can likewise be introduced into the linear predictor for detection. Both will account for the increased uncertainty in parameter estimates owing to the effects of unmodeled covariates by spreading out the posterior distributions and, hence, increasing uncertainty intervals. We see an example in [Section 12.3.3](#).

Third, the model assumes that all  $N_i$  individuals in each local population  $i$  behave and are detected independently. This assumption may be violated for animals that live in groups so that when one individual in a group is detected, others in the same group are more likely to be detected as well. An extension of the basic binomial mixture model to this situation has recently been described by Martin et al. (2011).

Fourth, the model as described so far is for static situations where replicate counts are available for a single, closed population of size  $N_i$  at each site  $i$ . This ideal will be impossible to attain in many situations, and indeed, in many cases, changes in abundance, for instance trends, are the focus of the modeling. The binomial mixture model can easily be extended to dynamic situations, for example, to several breeding seasons  $k$ , provided that data are available in the so-called robust design (Williams et al., 2002), with temporal replicate observations within each of multiple seasons. We then model counts  $y_{i,j,k}$  from site  $i$ , replicate  $j$ , and season  $k$  and estimate different parameters for each season in the ecological process and possibly also for the observation process. The abundance parameters may be related to each other across years, for instance, to model a trend (Royle and Dorazio, 2008; Kéry and Royle, 2010; Kéry et al., 2010a). We will see an example of this in [Section 12.3](#). In principle, it would also be possible to model dynamic (autoregressive) population models, such as the Ricker or Gompertz equations (Dennis et al., 2006), or the models in Chapter 5, within a binomial mixture framework, but this has not been done so far.

Key assumptions of the model are the following:

1. The ecological state is constant during the period over which replicate surveys are conducted (the traditional closure assumption). Its violation will lead to inflated abundance estimates. In benign cases, this may mean that  $N$  estimates the size of some superpopulation, that is, the number of all individuals that *ever* use a sample site during the surveys. In severe cases, though,  $N$  estimates may no longer be meaningful.
2. Detection probability is constant for all  $N_i$  individuals present at time  $j$  and equal to  $p_{i,j}$ . The model does not require individual identification but is not able to accommodate individual variation in detection probability either. In analogy to closed models (Chapter 6), intuition suggests that individual heterogeneity in detection probability at site  $i$  during period  $j$  leads to a negative bias in the abundance estimator (see Efford and Dawson (2009) for the special case of distance-related heterogeneity in detection).
3. The distribution of abundance  $N_i$  is adequately described by the chosen parametric form (e.g., a Poisson, possibly with covariates and latent effects). Similarly, detection probability is modeled adequately by the chosen parametric distribution (including possible covariates and other model structure). Especially the assumption about  $N_i$  is likely to be

more difficult to meet than the analogous assumption about occurrence in the site-occupancy models in Chapter 13. Effects of deviations of the data from the parametric model assumptions are hard to gauge, but posterior predictive checks (Section 12.3.) enable one to diagnose whether a model fits adequately.

4. There are no false positives such as double counts. The effect of the violation of this assumption has not been studied so far but is likely to induce a positive bias in the abundance estimator.

In Section 12.2, we first generate and analyze data from a single season (i.e., period of closure). In Section 12.3, we look at real-world data from multiple seasons in an insect species. A season may be one annual breeding season (for instance, for birds or reptiles, Kéry et al., 2005b, 2009a) or a single day, as in our insect example. We will fit a progression of increasingly complex models and see examples of zero-inflation and overdispersion correction. We will revisit (after their introduction in Chapter 7) an important and very general technique for goodness-of-fit assessment called a posterior predictive check (or Bayesian  $p$ -value).

## 12.2 GENERATION AND ANALYSIS OF SIMULATED DATA

---

### 12.2.1 The Simplest Case with Constant Parameters

To see the conceptual simplicity and beauty of the binomial mixture model, we first look at the simplest possible case, where both the ecological and the observation processes are described by an intercept only. To generate count data  $y$  under this Null model for  $R = 200$  spatial replicates (sites) and  $T = 3$  temporal replicates, we execute the following R commands:

```
# Determine sample sizes (spatial and temporal replication)
R <- 200
T <- 3

# Create structure to contain counts
y <- array(dim = c(R, T))

# Sample abundance from a Poisson(lambda = 2)
N <- rpois(n = R, lambda = 2)

# Sample counts from a Binomial(N, p = 0.5)
for (j in 1:T) {
  y[, j] <- rbinom(n = R, size = N, prob = 0.5)
}

# Look at realization of biological and observation processes
cbind(N, y)
```

We have assumed a mean abundance per site of 2 and mean detection per individual of 0.5. Note that in this model, the detection

parameter refers to each individual, whereas in the site-occupancy model (Chapter 13) it refers to the collection of *all* individuals inhabiting a site, that is, to an occupied site. Now, we will try to recover these parameter values when fitting the model in WinBUGS. Note how similar the BUGS code for this model is to the hierarchical way the data were created. Also, note that for this class of models, initial values for the latent states (the Ns) must sometimes be close to the solution; otherwise, WinBUGS may not achieve convergence or only do so very slowly. As our best guess, we choose the observed maximum count at each site, increased by 1, to save WinBUGS from having to update a Binomial with index 0.

```

# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
lambda ~ dgamma(0.005, 0.005)    # Standard vague prior for lambda
# lambda ~ dunif(0, 10)           # Other possibility
p ~ dunif(0, 1)

# Likelihood
# Biological model for true abundance
for (i in 1:R) {
  N[i] ~ dpois(lambda)
  # Observation model for replicated counts
  for (j in 1:T) {
    y[i,j] ~ dbin(p, N[i])
    } #j
  } #i
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(y = y, R = nrow(y), T = ncol(y))

# Initial values
Nst <- apply(y, 1, max) + 1 # This line is important
inits <- function() list(N = Nst)

# Parameters monitored
params <- c("lambda", "p")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT 0.1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

```

```
# Summarize posteriors
print(out, dig = 2)

      mean    sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
lambda 2.20 0.18  1.89  2.08  2.19  2.30  2.57  1.01    440
p       0.48 0.03  0.42  0.46  0.49  0.51  0.55  1.03    120
```

This looks good. There is a fair amount of sampling variance and so repeated realizations of the analysis will yield fairly variable estimates.

### 12.2.2 Introducing Covariates

Next, we simulate more complex data for a single season. We introduce a single covariate that acts on both the ecological and the observation process. We model the effect of the covariate on the scale of the log and the logit, as is customary for generalized linear models. This example illustrates a so-called site covariate, that is, a covariate that varies by site only, but not among individual surveys (this would be called a survey or sampling covariate). Sampling covariates may be weather condition or survey duration, that is, something that may affect detection but not abundance. For sampling covariates, see the exercises at the end of this chapter. OpenBUGS (Examples > Ecology examples > Lizards) also contains an example of a binomial mixture model with covariates for both abundance and detection.

```
# Define function for generating binom-mix model data
data.fn <- function(R = 200, T = 3, xmin = -1, xmax = 1, alpha0 = 1,
alpha1 = 3, beta0 = 0, beta1 = -5) {

# R: number of sites at which counts were made (= number of spatial reps)
# T: number of times that counts were made at each site
# (= number of temporal reps)
# xmin, xmax: define range of the covariate X
# alpha0 and alpha1: intercept and slope of log-linear regression
# relating abundance to the site covariate A
# beta0 and beta1: intercept and slope of logistic-linear regression
# of detection probability on A

y <- array(dim = c(R, T)) # Array for counts

# Ecological process
# Covariate values: sort for ease of presentation
X <- sort(runif(n = R, min = xmin, max = xmax))

# Relationship expected abundance - covariate
lam <- exp(alpha0 + alpha1 * X)

# Add Poisson noise: draw N from Poisson(lambda)
N <- rpois(n = R, lambda = lam)
table(N)                      # Distribution of abundances across sites
sum(N > 0) / R                # Empirical occupancy
totalN <- sum(N) ; totalN
```

```

# Observation process
# Relationship detection prob - covariate
p <- plogis(beta0 + betal * X)

# Make a 'census' (i.e., go out and count things)
for (i in 1:T) {
  y[,i] <- rbinom(n = R, size = N, prob = p)
}

# Naïve regression
naive.pred <- exp(predict(glm(apply(y, 1, max) ~ X + I(X^2),
  family = poisson)))

# Plot features of the simulated system
par(mfrow = c(2, 2))
plot(X, lam, main = "Expected abundance", xlab = "Covariate",
  ylab = "lambda", las = 1, type = "l", col = "red", lwd = 3,
  frame.plot = FALSE)
plot(X, N, main = "Realised abundance", xlab = "Covariate", ylab =
  "N", las = 1, frame.plot = FALSE, col = "red", cex = 1.2)
plot(X, p, ylim = c(0, 1), main = "Detection probability", xlab =
  "Covariate", ylab = "p", type = "l", col = "red", lwd = 3, las = 1,
  frame.plot = FALSE)
plot(X, naive.pred, main = "Actual counts \n and naive regression",
  xlab = "Covariate", ylab = "Relative abundance", ylim = c(min(y),
  max(y)), type = "l", lty = 2, lwd = 4, col = "blue", las = 1,
  frame.plot = FALSE)
points(rep(X, T), y, col = "black", cex = 1.2)

# Return stuff
return(list(R = R, T = T, X = X, alpha0 = alpha0, alphal = alphal,
  beta0 = beta0, betal = betal, lam = lam, N = N, totalN = totalN,
  p = p, y = y))
}

```

We execute this function once to generate one data set and produce an overview of the simulation.

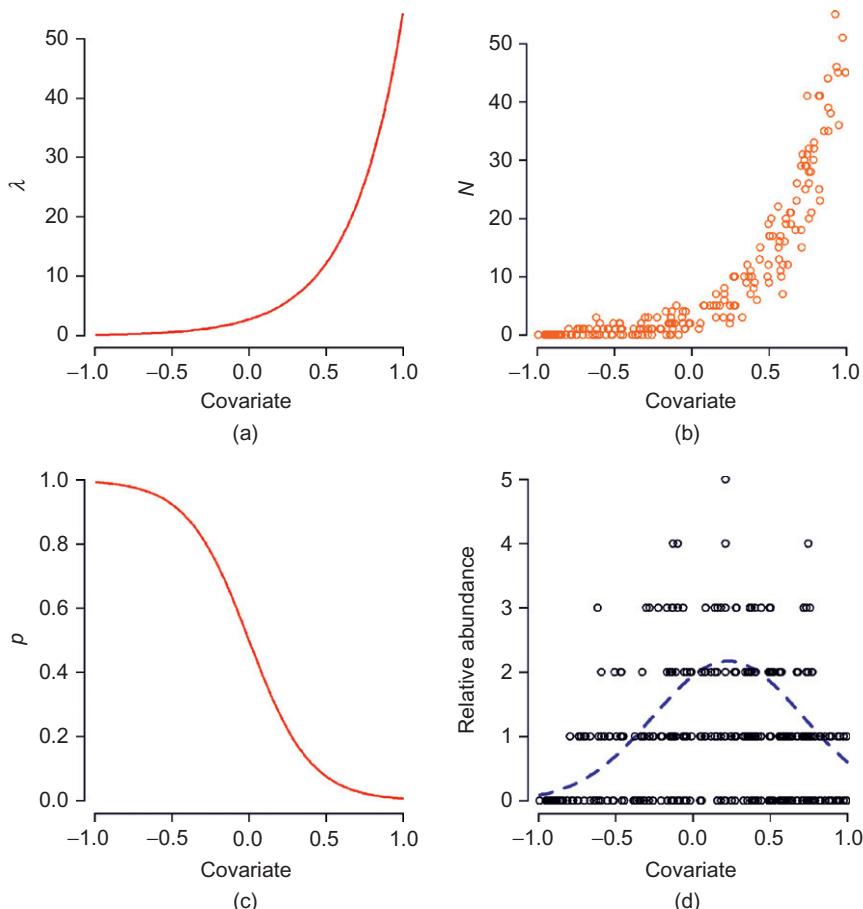
```

data <- data.fn()
str(data)

```

Figure 12.1 shows the main features of this stochastic system and its observation, that is, the data set generated. Abundance has a positive relation with the covariate (Fig. 12.1a and b). In contrast, detection probability has a negative relationship with that same covariate (Fig. 12.1c). The result of this is that the observed counts (Fig. 12.1d) suggest an intermediate optimum value of the covariate for abundance, and this false impression is confirmed by fitting a quadratic covariate effect in a Poisson regression of the max count at each site. The dashed blue line shows the prediction from that naïve analysis.

Next, we use a binomial mixture model to see whether we can tease apart the opposing effects of the covariate on the ecological state and on



**FIGURE 12.1** Features of the ecological and of the observation process that generated our data set and a conventional (naïve) analysis of counts in relation to an environmental covariate (dashed blue line in (d)): (a) Expected abundance, (b) realized abundance; (c) detection probability, (d) actual counts and naïve regression. The truth is shown in red and observed data in black. See text and R code for further explanations.

the observation of that state or, in plain English, whether we can recover estimates of the two GLM regressions that resemble the known input values. We are also interested in an estimate of the total population size across all surveyed plots, which, in our simulated data, was 1938.

```
# Specify model in BUGS language
sink("model.txt")
cat("
model {
```

```

# Priors
alpha0 ~ dunif(-10, 10)
alphal ~ dunif(-10, 10)
beta0 ~ dunif(-10, 10)
beta1 ~ dunif(-10, 10)

# Likelihood
# Ecological model for true abundance
for (i in 1:R){
  N[i] ~ dpois(lambda[i])
  log(lambda[i]) <- alpha0 + alphal * X[i]

  # Observation model for replicated counts
  for (j in 1:T){
    y[i,j] ~ dbin(p[i,j], N[i])
    p[i,j] <- exp(lp[i,j])/(1+exp(lp[i,j]))
    lp[i,j] <- beta0 + beta1 * X[i]
  } #j
} #i

# Derived quantities
totalN <- sum(N[])
}

",fill = TRUE)
sink()

# Bundle data
y <- data$y
win.data <- list(y = y, R = nrow(y), T = ncol(y), X = data$X)

# Initial values
Nst <- apply(y, 1, max) + 1 # Important to give good inits for latent N
inits <- function() list(N = Nst, alpha0 = runif(1, -1, 1), alphal = runif(1, -1, 1),
  beta0 = runif(1, -1, 1), beta1 = runif(1, -1, 1))

# Parameters monitored
params <- c("totalN", "alpha0", "alphal", "beta0", "beta1")

# MCMC settings
ni <- 22000
nt <- 20
nb <- 2000
nc <- 3

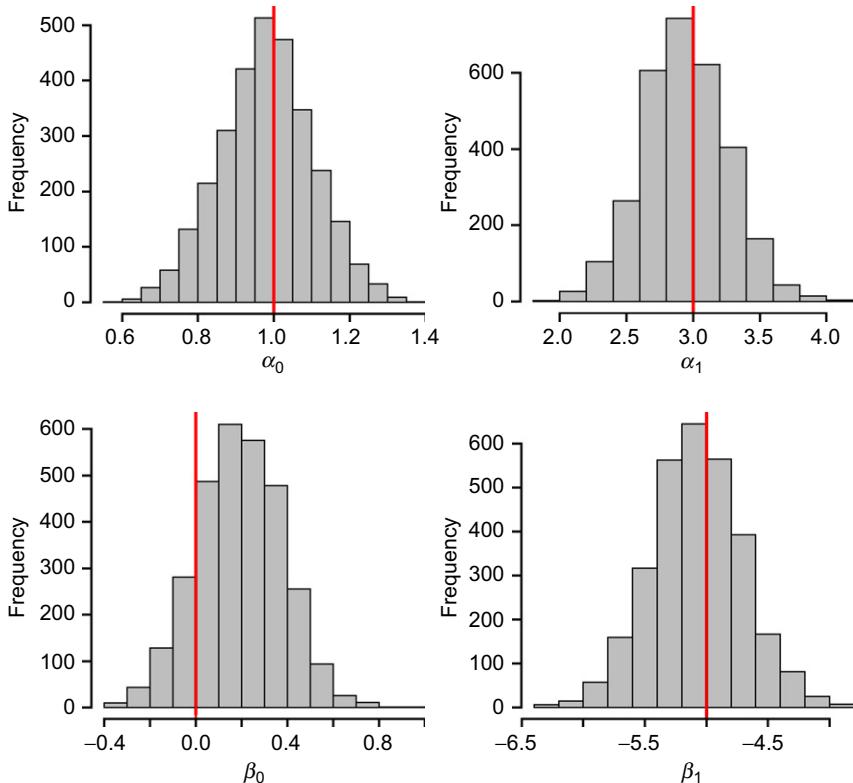
# Call WinBUGS from R (BRT 4 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out, dig = 3)
      mean      sd     2.5%     25%     50%     75%   97.5%   Rhat   n.eff
totalN    1866.708  574.032   982.975  1464.750  1759.000  2192.000  3214.225  1.068      37
alpha0      0.981    0.123     0.736     0.900     0.983     1.064     1.223    1.027      83

```

alpha1	2.941	0.319	2.303	2.726	2.931	3.151	3.575	1.060	39
beta0	0.191	0.187	-0.178	0.064	0.192	0.319	0.551	1.023	98
beta1	-5.078	0.370	-5.806	-5.328	-5.078	-4.834	-4.339	1.040	58
deviance	853.819	14.566	826.897	843.875	853.000	863.300	883.700	1.013	160

We recover estimates that appear unbiased. The estimate of total population size is fairly imprecise, but will often, and here does, contain the true value (1938) within its 95% CRI. The sum of the maximum counts at each site, a conventional estimator of total population size, is only 240 individuals (type `sum(apply(data$y, 1, max))`). It is remarkable that the binomial mixture model gets so close to the truth! Let us look at further inferences and plot the posterior distribution for the four regression parameters (Fig. 12.2).

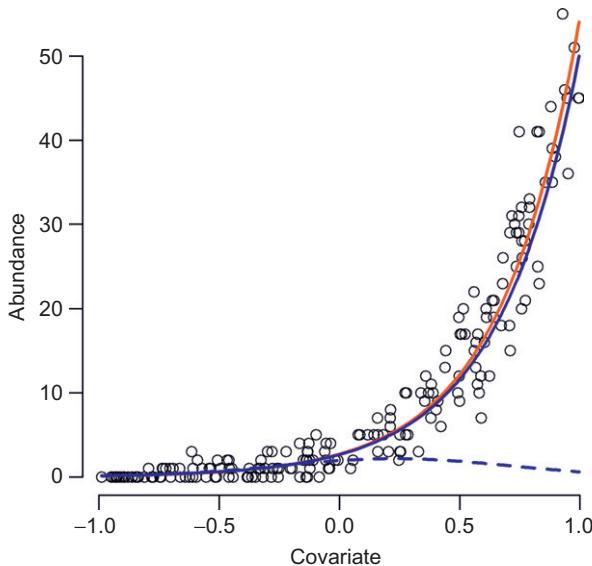


**FIGURE 12.2** Posterior distributions for the two log- and logit-linear regressions of expected abundance and detection probability on a covariate. The red lines show the values used for these parameters in the simulation of the data.

Typically, an ecologist would want estimates of abundance for each site also. Within the Bayesian framework, this is trivial: just add  $N$  to the list of quantities monitored!

```
# Plot posteriors
par(mfrow = c(2, 2))
hist(out$sims.list$alpha0, col = "gray", main = "", xlab = "alpha0",
     las = 1)
abline(v = data$alpha0, lwd = 3, col = "red")
hist(out$sims.list$alpha1, col = "gray", main = "", xlab = "alpha1",
     las = 1)
abline(v = data$alpha1, lwd = 3, col = "red")
hist(out$sims.list$beta0, col = "gray", main = "", xlab = "beta0", las=1)
abline(v = data$beta0, lwd = 3, col = "red")
hist(out$sims.list$beta1, col = "gray", main = "", xlab = "beta1", las = 1)
abline(v = data$beta1, lwd = 3, col = "red")
```

How well can the naive and the binomial mixture analysis recover the covariate relationship? Figure 12.3 shows that for the conventional model, the answer is: not very well! In contrast, the binomial mixture model seems to do an excellent job at recovering the true positive relationship between abundance and the covariate.



**FIGURE 12.3** Relationship between abundance and covariate in a simulated system. The red line shows truth, with realized abundance at 200 sites in black. The blue lines show the estimates under a binomial mixture model (solid line) and under a conventional nonhierarchical Poisson regression (dashed).

```
# Plot predicted covariate relationship with abundance
plot(data$X, data$N, main = "", xlab = "Covariate", ylab = "Abundance",
      las = 1, ylim = c(0, max(data$N)), frame.plot = FALSE)
lines(data$X, data$lam, type = "l", col = "red", lwd = 3)
GLM.pred <- exp(predict(glm(apply(data$y, 1, max) ~ X + I(X^2),
      family = poisson, data = data)))
lines(data$X, GLM.pred, type = "l", lty = 2, col = "blue", lwd = 3)
Nmix.pred <- exp(out$mean$alpha0 + out$mean$alphal * data$X)
points(data$X, Nmix.pred, type = "l", col = "blue", lwd = 3)
```

## 12.3 ANALYSIS OF REAL DATA: OPEN-POPULATION BINOMIAL MIXTURE MODELS

Frequently, counts are made at two temporal scales. That is, one does not only have count data from a single period of closure, but in addition from several seasons. For instance, for birds, we may have repeated samples within a breeding season (visits within a year are secondary occasions) that are repeated across several years (years are primary occasions). In capture–recapture, this sampling design is called the robust design (Williams et al., 2002): a population is repeatedly sampled over a short period, during which the population is considered to be closed, and these sampling sessions are repeated over a longer period, over which the population is considered to be open. The primary occasions may also consist of (much) shorter time intervals than years if the population dynamics of the study organism is fast relative to the duration of a study.

In our real-world example, we will estimate population size of a butterfly, the silver-washed fritillary (Fig. 12.4), over an entire summer. The information for application of the binomial mixture model comes from the fact that each of 95 sites was surveyed twice along a 2.5 km transect on each of 4–7 survey days (Kéry et al., 2009a; Dorazio et al., 2010). Survey days are separated by at least two weeks, and butterflies are rather short-lived; hence, it would not be sensible to assume a constant population size at each site over repeated days or even during the entire summer. Hence, we shall consider each day a primary occasion and model different parameters for each day. The two replicate observations of each transects within a day represent the secondary occasions between which the populations are assumed constant. We will apply the open-population binomial mixture model of Dorazio and Royle (2008), Kéry et al. (2009a), and Kéry and Royle (2010) to estimate population size during each day.

We denote as  $y_{ijk}$  the count from site  $i$  ( $i = 1 \dots 95$ ), within-day temporal replicate  $j$  ( $j = 1, 2$ ), and day  $k$  ( $k = 1 \dots 7$ ). Note that both  $j$  and  $k$  index temporal replicates, but the population is assumed to be static over the former and allowed to change over the latter. In program R, we format



**FIGURE 12.4** Silver-washed fritillary *Argynnis paphia*, Switzerland, 2005 (Photograph by T. Marent).

the counts into a three-dimensional array, since this is convenient for the modeling. In our experience, one of the most difficult things about statistical modeling in WinBUGS is to keep track of the dimensions of multi-dimensional arrays, including putting data into such arrays in the first place. So, expect some time to learn how to do this, either in a clumsy way like we do here or using R functions like those in the `reshape` package.

We will revisit posterior predictive distributions and Bayesian  $p$ -values in this example, introduced in Section 7.10, a very general method of checking the goodness-of-fit of a model fit using Bayesian posterior sampling (see Gelman et al., 2004; Gelman et al., 1996, Kéry, 2010). The posterior predictive distribution is the distribution of replicate data sets or statistics computed from data sets, given the model, its parameter values, and the observed data set. We do a posterior predictive check of whether the model used to analyze the observed data fits them, in the sense that the model, with the estimated parameters, could plausibly have generated data such as the data set that we actually observed. The idea behind a posterior predictive check is rather similar to a parametric bootstrap: use the parameter values estimated from the actual data set under a given model to generate new data sets. Then, calculate some discrepancy measure for the new data set (e.g., chi-squared) and repeat that many times to get the reference distribution for that discrepancy measure for a model

that fits. Then, compare the fit of the model with the actual data with that reference distribution. For example, we see whether a, say, chi-squared test statistic computed for the actual data set falls within the distribution of chi-squared statistics that was computed for the replicate data sets.

We load the data, put them into a 3D array, and look at some summary statistics.

```
# Get the data and put them into 3D array
bdat <- read.table("fritillary.txt", header = TRUE)
y <- array(NA, dim = c(95, 2, 7)) # 95 sites, 2 reps, 7 days

for(k in 1:7) {
  sel.rows <- bdat$day == k
  y[, , k] <- as.matrix(bdat)[sel.rows, 3:4]
}
y # Look at data set in 3D layout
str(y)

# Have a look at raw data
day.max <- apply(y, c(1, 3), max, na.rm = TRUE) # Max count each site
                                                    and day
day.max
site.max <- apply(day.max, 1, max, na.rm = TRUE) # Max count each site
site.max
table(site.max) # Frequency distribution of max counts
plot(table(site.max))
table(site.max>0) # Observed occupancy is only 56%

# Sum of observed max as conventional estimator of total abundance
max1 <- apply(y, c(1, 3), max)
obs.max.sum <- apply(max1, 2, sum, na.rm = TRUE)

obs.max.sum
[1] 4 0 15 32 99 85 63
```

Very few butterflies were observed during the first day and none during the second day.

### 12.3.1 Simple Poisson Model

We start with the simplest binomial mixture model that appears to make sense in this case. It has time-specific, constant parameters for both abundance and detection probability. Note that in the computation of the chi-squared discrepancy measure for the posterior predictive check, a small constant (0.5) is added in the denominator to avoid possible divisions by zero.

```
# Specify model in BUGS language
sink("Nmix0.txt")
cat("
model {
```

```

# Priors
for (k in 1:7) {
  alpha.lam[k] ~ dnorm(0, 0.01)
  p[k] ~ dunif(0, 1)
}

# Likelihood
# Ecological model for true abundance
for (k in 1:7) {                                     # Loop over days (7)
  lambda[k] <- exp(alpha.lam[k])
  for (i in 1:R){                                 # Loop over R sites (95)
    N[i,k] ~ dpois(lambda[k])                   # Abundance

  # Observation model for replicated counts
  for (j in 1:T){                               # Loop over temporal reps (2)
    y[i,j,k] ~ dbin(p[k], N[i,k])             # Detection

    # Assess model fit using Chi-squared discrepancy
    # Compute fit statistic E for observed data
    eval[i,j,k] <- p[k] * N[i,k]               # Expected values
    E[i,j,k] <- pow((y[i,j,k] - eval[i,j,k]),2) / (eval[i,j,k] +
      0.5)
    # Generate replicate data and compute fit stats for them
    y.new[i,j,k] ~ dbin(p[k], N[i,k])
    E.new[i,j,k] <- pow((y.new[i,j,k] - eval[i,j,k]),2) /
      (eval[i,j,k] + 0.5)

    } #j
  } #i
} #k

# Derived and other quantities
for (k in 1:7) {
  totalN[k] <- sum(N[,k])          # Total pop. size across all sites
  mean.abundance[k] <- exp(alpha.lam[k])
}
fit <- sum(E[,,])
fit.new <- sum(E.new[,,])
}
",fill = TRUE)
sink()

# Bundle data
R = nrow(y)
T = ncol(y)
win.data <- list(y = y, R = R, T = T)

# Initial values
Nst <- apply(y, c(1, 3), max) + 1
Nst[is.na(Nst)] <- 1
inits <- function(){list(N = Nst, alpha.lam = runif(7, -1, 1))}

# Parameters monitored
params <- c("totalN", "mean.abundance", "alpha.lam", "p", "fit",
  "fit.new")

```

```

# MCMC settings
ni <- 10000
nt <- 8
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
out0 <- bugs(win.data, inits, params, "Nmix0.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out0, dig = 3)

      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
totalN[1] 35.133 59.426  4.000  8.000 15.000 32.000 233.148 1.090    49
totalN[2]  0.162  0.902  0.000  0.000  0.000  0.000  2.000 1.132   330
totalN[3] 20.003  5.097 15.000 17.000 19.000 22.000 34.000 1.002 1800
totalN[4] 39.511  5.117 33.000 36.000 38.000 42.000 52.000 1.003 2400
totalN[5] 135.943 13.689 115.000 126.000 134.000 144.000 168.000 1.001 3000
totalN[6] 99.653  6.869 90.000 95.000 99.000 103.000 116.000 1.004 750
totalN[7] 90.296 10.605 74.000 83.000 89.000 96.000 115.000 1.001 3000
mean.abundance[1] 0.371  0.631  0.030  0.088  0.163  0.348  2.447 1.077    56
mean.abundance[2] 0.003  0.010  0.000  0.000  0.000  0.001  0.023 1.001 3000
mean.abundance[3] 0.211  0.071  0.109  0.165  0.201  0.245  0.389 1.002 1900
mean.abundance[4] 0.415  0.086  0.271  0.356  0.407  0.465  0.608 1.001 3000
mean.abundance[5] 1.430  0.191  1.091  1.299  1.415  1.549  1.843 1.001 3000
mean.abundance[6] 1.048  0.128  0.820  0.960  1.040  1.127  1.316 1.003 930
mean.abundance[7] 0.952  0.151  0.686  0.850  0.939  1.041  1.277 1.001 3000
[...]
p[1]        0.212  0.167  0.012  0.077  0.170  0.312  0.625 1.070    60
p[2]        0.485  0.291  0.017  0.229  0.489  0.731  0.972 1.003 1300
p[3]        0.545  0.124  0.282  0.465  0.552  0.634  0.766 1.002 3000
p[4]        0.600  0.087  0.416  0.543  0.605  0.661  0.761 1.002 1700
p[5]        0.515  0.057  0.401  0.475  0.518  0.556  0.619 1.001 3000
p[6]        0.653  0.054  0.539  0.619  0.656  0.692  0.750 1.002 1500
p[7]        0.550  0.068  0.416  0.505  0.552  0.597  0.677 1.001 3000
fit       147.924 15.887 121.000 136.675 146.400 157.625 183.102 1.001 3000
fit.new    97.796 11.908 76.427 89.490 97.220 104.900 123.000 1.002 3000
deviance  765.186 42.162 689.300 735.575 763.400 792.200 855.505 1.001 2700

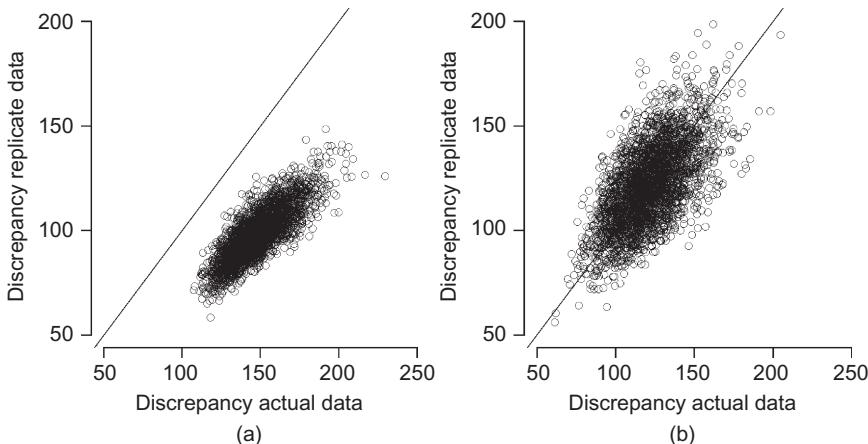
```

We note that the estimates for the first two days, when very few or no butterflies were observed, are very imprecise and entirely driven by the priors, respectively: the 95% CRI for  $p$  at day 2 extends almost from 0 to 1. We look at the posterior predictive check of goodness-of-fit for this model (Fig. 12.5a).

```

# Evaluation of fit
plot(out0$sims.list$fit, out0$sims.list$fit.new, main = "", xlab =
  "Discrepancy actual data", ylab = "Discrepancy replicate data",
  frame.plot = FALSE)
abline(0, 1, lwd = 2, col = "black")

```



**FIGURE 12.5** Posterior predictive checks of model adequacy of two binomial mixture models fit to the Swiss fritillary data; (a) Null model, (b) model with random effects in abundance and detection (see Section 12.3.3.).

```
mean(out0$sims.list$fit.new > out0$sims.list$fit)
[1] 0
mean(out0$mean$fit) / mean(out0$mean$fit.new)
[1] 1.512573
```

The model does not fit well at all (Fig. 12.5a). Indeed, there is a “lack-of-fit ratio” of 1.51. This informal quantity compares the mean of the fit statistic for the actual data with that for the perfect data sets and thus gives a numerical expression of how bad the lack of fit is. What could be wrong? The fritillary was never detected at 42 sites and detected at least once at 53; yielding an observed occupancy of 56%. It may well be that a Poisson distribution for the spatial variation in abundance is not flexible enough to account for that many zeroes. Therefore we will next model abundance with a zero-inflated Poisson distribution (Wenger and Freeman, 2008; Joseph et al., 2009).

### 12.3.2 Zero-Inflated Poisson Binomial Mixture Model (ZIP Binomial Mixture Model)

The ZIP binomial mixture model appears like a sensible model for these data. It divides the sites into those that are suitable and those that are not suitable. A Poisson distribution for abundance is assumed for suitable sites only. To obtain a ZIP binomial mixture model, we simply add another hierarchical layer to the model. This additional layer is binary: we decide by a coin flip whether a site is suitable in principle or not. Only if a site is suitable in principle will Nature roll her Poisson die

to determine the actual number of butterflies living there. Here is the resulting hierarchical model:

$$\begin{aligned} \text{Level 1 (suitability of site } i\text{):} & \quad z_i \sim \text{Bernoulli}(\Omega) \\ \text{Level 2 (realized abundance at } i, k\text{):} & \quad N_{i,k} | z_i \sim \text{Poisson}(z_i \lambda_k) \\ \text{Level 3 (observed count at } i, j, k\text{):} & \quad y_{i,j,k} | N_{i,k} \sim \text{Binomial}(N_{i,k}, p_{i,j,k}) \end{aligned}$$

We have three main structural parameters ( $\Omega$ ,  $\lambda_k$ ,  $p_{i,j,k}$ ), one for each level in the model hierarchy. We could model each of them as a function of covariates through a GLM link function. In our case, we do not have any covariates, so we simply fit group effects, that is, estimate a separate parameter for abundance and detection at every time period.

To describe the model in the BUGS language, we want to define the latent suitability indicators  $z$  first, that is, in the outermost loop of the likelihood definition. Thus, we simply flip the order in which we loop over the dimensions of the site-by-rep-by-day data array  $y_{i,j,k}$ .

```
# Specify model in BUGS language
sink("Nmix1.txt")
cat("
model {

# Priors
omega ~ dunif(0, 1)
for (k in 1:7){
  alpha.lam[k] ~ dnorm(0, 0.01)
  p [k] ~ dunif(0, 1)
}

# Likelihood
# Ecological model for true abundance
for (i in 1:R){                                # Loop over R sites (95)
  z[i] ~ dbern(omega)                         # Latent suitability state
  for (k in 1:7){                            # Loop over survey periods (seasons)
    N[i,k] ~ dpois(lam.eff[i,k]) # Latent abundance state
    lam.eff[i,k] <- z[i] * lambda[i,k]
    log(lambda[i,k]) <- alpha.lam[k]

# Observation model for replicated counts
  for (j in 1:T){                            # Loop over temporal reps (2)
    y[i,j,k] ~ dbin(p[k], N[i,k])   # Detection
    # Assess model fit using Chi-squared discrepancy
    # Compute fit statistic for observed data
    eval[i,j,k] <- p[k] * N[i,k]
    E[i,j,k] <- pow((y[i,j,k] - eval[i,j,k]), 2) / (eval[i,j,k] +
      0.5)
    # Generate replicate data and compute fit stats for them
    y.new[i,j,k] ~ dbin(p[k], N[i,k])
    E.new[i,j,k] <- pow((y.new[i,j,k] - eval[i,j,k]), 2) /
      (eval[i,j,k]+0.5)
  } #j
} #k
} #i
```

```

# Derived and other quantities
for (k in 1:7) {
  # Estimate total pop. size across all sites
  totalN[k] <- sum(N[,k])
  mean.abundance[k] <- exp(alpha.lam[k])
}
fit <- sum(E[,])
fit.new <- sum(E.new[,])
},
",fill = TRUE)
sink()

# Bundle data
R = nrow(y)
T = ncol(y)
win.data <- list(y = y, R = R, T = T)

# Initial values
Nst <- apply(y, c(1, 3), max) + 1
Nst[is.na(Nst)] <- 1
inits <- function() {list(N = Nst, alpha.lam = runif(7, -1, 1))}

# Parameters monitored
params <- c("omega", "totalN", "alpha.lam", "p", "mean.abundance",
  "fit", "fit.new")

# MCMC settings
ni <- 30000
nt <- 15
nb <- 15000
nc <- 3

# Call WinBUGS from R (BRT 3 min)
out1 <- bugs(win.data, inits, params, "Nmix1.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out1, dig = 3)
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat   n.eff
omega    0.561  0.050   0.464   0.527   0.561   0.594   0.655 1.001   3000
totalN[1] 21.063 31.485   4.000   7.000  11.000  21.000  99.025 1.050    57
totalN[2]  0.145  0.756   0.000   0.000   0.000   0.000   2.000 1.068   300
totalN[3] 21.074  6.506  15.000  17.000  19.000  23.000  38.000 1.005  1000
totalN[4] 40.953  7.002  33.000  36.747  39.000  43.000  59.000 1.003  1900
totalN[5] 154.783 23.744 122.000 139.000 151.000 166.000 212.000 1.003  760
totalN[6] 105.251  9.486  91.975  99.000 104.000 110.000 128.000 1.002  3000
totalN[7]  91.561 14.688  72.000  81.000  89.000  98.000 129.025 1.002  2500
[...]
p[1]      0.235  0.170   0.017   0.096   0.197   0.342   0.647 1.043    63
p[2]      0.479  0.295   0.016   0.212   0.474   0.736   0.970 1.001  3000
p[3]      0.527  0.132   0.262   0.437   0.535   0.624   0.766 1.006    580
p[4]      0.585  0.094   0.384   0.526   0.591   0.651   0.748 1.001  3000
p[5]      0.458  0.067   0.322   0.415   0.459   0.504   0.582 1.003  1000
p[6]      0.621  0.060   0.496   0.582   0.623   0.665   0.725 1.001  3000
p[7]      0.503  0.079   0.338   0.452   0.507   0.558   0.651 1.001  2400
[...]

```

```
fit      147.329 16.573 119.900 135.700 145.800 157.000 184.500 1.003 780
fit.new   97.575 11.791  76.604  89.475  96.855 105.000 122.902 1.003 990
deviance 751.830 38.437 680.582 725.000 749.800 775.825 834.300 1.004 610
```

Under this model, the deviance is somewhat improved (here, from 765 to 752). The proportion of suitable sites,  $\omega$ , is estimated identically with the proportion of sites at which the butterfly was ever detected. But does this model fit?

```
# Evaluation of fit
plot(out1$sims.list$fit, out1$sims.list$fit.new, main = "", xlab =
  "Discrepancy actual data", ylab = "Discrepancy replicate data",
  frame.plot = FALSE)
abline(0, 1, lwd = 2, col = "black")
mean(out1$sims.list$fit.new > out1$sims.list$fit)
[1] 0
mean(out1$mean$fit) / mean(out1$mean$fit.new)
[1] 1.509898
```

Unfortunately, the model still does not fit the data according to our chosen discrepancy measure, which is much greater for the actual data set than for the replicate data sets (i.e., the reference distribution of the test statistic). The Bayesian  $p$ -value, the proportion of symbols above the 1:1 line in the figure, is equal to zero. Compared with the model without zero-inflation, the lack-fit-ratio has barely gone down (from 1.513 to 1.510). Hence, a zero-inflated binomial mixture model is not flexible enough to capture the variability in the system adequately.

### 12.3.3 Binomial Mixture Model with Overdispersion in Both Abundance and Detection

Finally, we drop zero-inflation and instead try a model that accounts for extra-Poisson dispersion in both abundance and detection. The introduction of latent (random) effects into either or both linear predictors can be seen as sort of overdispersion correction, and it increases the uncertainty in the estimates. Thus, as in overdispersion correction in capture–recapture for instance (Burnham and Anderson, 2002), we buy a fitting model by losing precision in our estimates.

Level 1 (realized abundance at  $i, k$ ):  $N_{i,k} \sim \text{Poisson}(\lambda_{i,k})$

GLM for level 1:  $\log(\lambda_{i,k}) = \alpha_k + \varepsilon_i$

Level 1b (random site effects):  $\varepsilon_i \sim \text{Normal}(0, \sigma_\lambda^2)$

Level 2 (observed count at  $i, j, k$ ):  $y_{i,j,k} | N_{i,k} \sim \text{Binomial}(N_{i,k}, p_{i,j,k})$

$$\text{GLM for level 2:} \quad \text{logit}(p_{i,j,k}) = \beta_k + \delta_{i,j,k}$$

$$\text{Level 2b (random survey effects):} \quad \delta_{i,j,k} \sim \text{Normal}(0, \sigma_p^2)$$

So, we combine a naturally hierarchical model with two additional hierarchical levels, one for the sites in the ecological process and another for the individual surveys in the observation process. We assume the presence of “residual” site- and site-day-replicate-specific contributions to abundance ( $\varepsilon$ ) and detection probability ( $\delta$ ), respectively. By specifying a Poisson-log-normal (Millar, 2009) model for the ecological state description, we account for the fact that there may be extra-Poisson dispersion in the distribution used to model spatio-temporal variation in the latent abundance parameters. Similarly, observation conditions may vary among sites, days, and replicates, and the random effect  $\delta_{i,j,k}$  adequately accounts for that additional variability. As is customary for this type of modeling, we assume a normal distribution for both random noise terms. Note that for the extra variability in detection (delta), we could model  $\delta_{i,j,k}$ ,  $\delta_{i,k}$ , or  $\delta_i$ . Our treatment here is consistent with the published analyses in Kéry et al. (2009a) and Kéry and Royle (2010) as well as with an unpublished study by L. Tanadini and M. Kéry.

```
# Specify model in BUGS language
sink("Nmix2.txt")
cat("
model{

# Priors
for (k in 1:7) {
    alpha.lam[k] ~ dnorm(0, 0.1)
    beta[k] ~ dnorm(0, 0.1)
}

# Abundance site and detection site-by-day random effects
for (i in 1:R) {
    eps[i] ~ dnorm(0, tau.lam)                                # Abundance noise
}
tau.lam <- 1 / (sd.lam * sd.lam)
sd.lam ~ dunif(0, 3)
tau.p <- 1 / (sd.p * sd.p)
sd.p ~ dunif(0, 3)

# Likelihood
# Ecological model for true abundance
for (i in 1:R){                                              # Loop over R sites (95)
    for (k in 1:7){                                         # Loop over days (7)
        N[i,k] ~ dpois(lambda[i,k])                         # Abundance
        log(lambda[i,k]) <- alpha.lam[k] + eps[i]
    }
}

# Observation model for replicated counts
for (j in 1:T){                                              # Loop over temporal
    rep[1] <- rbinom(1, N[1,rep[1]], p[1])
    rep[2] <- rbinom(1, N[2,rep[2]], p[2])
}
```

```

y[i,j,k] ~ dbin(p[i,j,k], N[i,k])    # Detection
p[i,j,k] <- 1 / (1 + exp(-lp[i,j,k]))
lp[i,j,k] ~ dnorm(beta[k], tau.p)    # Random delta defined
                                         implicitly

# Assess model fit using Chi-squared discrepancy
# Compute fit statistic for observed data
eval[i,j,k] <- p[i,j,k] * N[i,k]
E[i,j,k] <- pow((y[i,j,k] - eval[i,j,k]),2) / (eval[i,j,k]
+0.5)
# Generate replicate data and compute fit stats for them
y.new[i,j,k] ~ dbin(p[i,j,k], N[i,k])
E.new[i,j,k] <- pow((y.new[i,j,k] - eval[i,j,k]),2) /
(eval[i,j,k]+0.5)
} #j
ik.p[i,k] <- mean(p[i,,k])
} #k
} #i

# Derived and other quantities
for (k in 1:7) {
  totalN[k] <- sum(N[,k])      # Estimate total pop. size across all
  sites
  mean.abundance[k] <- mean(lambda[,k])
  mean.N[k] <- mean(N[,k])
  mean.detection[k] <- mean(ik.p[,k])
}
fit <- sum(E[,])
fit.new <- sum(E.new[,])
}
",fill = TRUE)
sink()

# Bundle data
R = nrow(y)
T = ncol(y)
win.data <- list(y = y, R = R, T = T)

# Initial values
Nst <- apply(y, c(1, 3), max) + 1
Nst[is.na(Nst)] <- 1
inits <- function(){list(N = Nst, alpha.lam = runif(7, -3, 3), beta =
runif(7, -3, 3), sd.lam = runif(1, 0, 1), sd.p = runif(1, 0, 1))}

# Parameters monitored
params <- c("totalN", "alpha.lam", "beta", "sd.lam", "sd.p",
"mean.abundance", "mean.N", "mean.detection", "fit", "fit.new")

```

Models with lots of random effects always need much longer to enable the Markov chains to mix properly. Hence, we greatly increase the number of MCMC iterations (obviously, we did this after some initial experimenting). We also increase the thinning rate to avoid having to save huge results files.

```
# MCMC settings
ni <- 350000
nt <- 300
nb <- 50000
nc <- 3
# Call WinBUGS from R (BRT 215 min)
out2 <- bugs(win.data, inits, params, "Nmix2.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())
```

We get done after almost 4 h. First, we evaluate the fit of the model (Fig. 12.5b): it does fit now!

```
# Evaluation of fit
plot(out2$sims.list$fit, out2$sims.list$fit.new, main = "", xlab =
  "Discrepancy actual data", ylab = "Discrepancy replicate data",
  frame.plot = FALSE, xlim = c(50, 200), ylim = c(50, 200))
abline(0, 1, lwd = 2, col = "black")
mean(out2$sims.list$fit.new > out2$sims.list$fit)
[1] 0.505
mean(out2$mean$fit) / mean(out2$mean$fit.new)
[1] 0.999935

# Summarize posteriors
print(out2, dig = 2)
```

	mean	sd	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
totalN[1]	94.48	163.99	5.00	13.00	35.00	95.00	593.05	1.01	370
totalN[2]	120.22	461.11	0.00	0.00	2.00	18.00	1762.90	1.36	25
totalN[3]	225.13	381.69	19.00	45.00	101.00	234.25	1537.05	1.06	55
totalN[4]	55.89	26.21	34.00	41.00	48.00	61.00	126.02	1.00	1100
totalN[5]	830.35	575.54	204.97	395.00	640.50	1119.25	2353.15	1.02	180
totalN[6]	128.36	30.90	96.00	109.00	121.00	138.00	209.00	1.01	390
totalN[7]	158.64	86.42	83.00	107.00	131.00	178.00	402.05	1.00	630
[...]									
sd.lam	1.87	0.23	1.46	1.70	1.84	2.01	2.37	1.00	1000
sd.p	1.05	0.21	0.70	0.91	1.03	1.17	1.50	1.00	980
mean.abundance[1]	1.00	1.73	0.04	0.14	0.37	1.00	6.30	1.01	340
mean.abundance[2]	1.27	4.86	0.00	0.01	0.03	0.18	18.73	1.07	48
mean.abundance[3]	2.37	4.01	0.18	0.48	1.07	2.46	16.44	1.06	55
mean.abundance[4]	0.59	0.29	0.31	0.43	0.52	0.66	1.38	1.00	1100
mean.abundance[5]	8.72	6.06	2.10	4.16	6.78	11.68	24.91	1.02	190
mean.abundance[6]	1.35	0.35	0.93	1.14	1.29	1.48	2.18	1.01	420
mean.abundance[7]	1.67	0.92	0.83	1.13	1.38	1.88	4.32	1.00	760
mean.N[1]	0.99	1.73	0.05	0.14	0.37	1.00	6.24	1.01	370
mean.N[2]	1.27	4.85	0.00	0.00	0.02	0.19	18.55	1.36	25
mean.N[3]	2.37	4.02	0.20	0.47	1.06	2.47	16.18	1.06	55
mean.N[4]	0.59	0.28	0.36	0.43	0.51	0.64	1.33	1.00	1100
mean.N[5]	8.74	6.06	2.16	4.16	6.74	11.78	24.77	1.02	180
mean.N[6]	1.35	0.33	1.01	1.15	1.27	1.45	2.20	1.01	390
mean.N[7]	1.67	0.91	0.87	1.13	1.38	1.87	4.23	1.00	630
mean.detection[1]	0.12	0.14	0.00	0.02	0.06	0.16	0.49	1.01	310

mean.detection[2]	0.23	0.32	0.00	0.01	0.04	0.39	0.99	1.06	62
mean.detection[3]	0.14	0.14	0.01	0.04	0.08	0.19	0.52	1.06	55
mean.detection[4]	0.47	0.14	0.18	0.38	0.48	0.56	0.71	1.01	810
mean.detection[5]	0.14	0.08	0.03	0.07	0.12	0.18	0.34	1.02	200
mean.detection[6]	0.51	0.10	0.29	0.45	0.52	0.58	0.69	1.01	460
mean.detection[7]	0.34	0.11	0.12	0.27	0.35	0.43	0.55	1.00	610
fit	121.41	18.67	88.48	108.60	120.20	133.22	161.41	1.01	240
fit.new	121.42	19.04	86.02	108.07	120.60	134.20	161.40	1.01	240
deviance	640.44	49.86	540.00	607.37	641.30	674.42	737.21	1.01	250

We note that convergence for some quantities associated with day 2 (when no fritillaries were observed at all) is less good. This illustrates the fact that often, though not always, lack of identifiability is associated with lack of convergence.

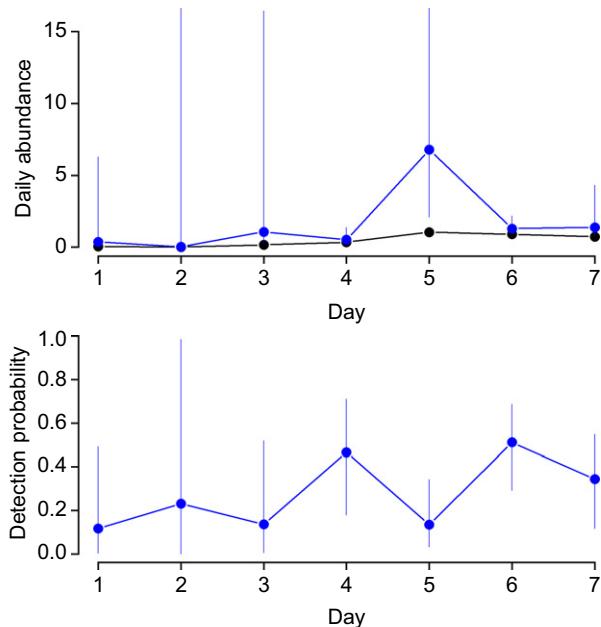
Since we have a fitting model now, we produce some plots of the estimates. In particular, we compare the mean count per day with the estimated mean abundance of the fritillary on each day. For mean daily abundance, we use the posterior median as our measure of central tendency, since the posterior is highly skewed (you can check that by typing `hist(out2$sims.list$mean.abundance[, 4], breaks = 40)`).

```
max.day.count <- apply(y, c(1, 3), max, na.rm = TRUE)
max.day.count [max.day.count == "-Inf"] <- NA
mean.max.count <- apply(max.day.count, 2, mean, na.rm = TRUE)
mean.max.count

par(mfrow = c(2, 1))
plot(1:7, mean.max.count, xlab = "Day", ylab = "Mean daily abundance",
  las = 1, ylim = c(0, 16), type = "b", main = "", frame.plot = FALSE,
  pch = 16, lwd = 2)
lines(1:7, out2$summary[24:30, 5], type = "b", pch = 16, col = "blue",
  lwd = 2)
segments(1:7, out2$summary[24:30, 3], 1:7, out2$summary[24:30, 7],
  col = "blue")

plot(1:7, out2$summary[38:44, 1], xlab = "Day", ylab = "Detection
  probability ", las = 1, ylim = c(0, 1), type = "b", col = "blue",
  pch = 16, frame.plot = FALSE, lwd = 2)
segments(1:7, out2$summary[38:44, 3], 1:7, out2$summary[38:44, 7],
  col = "blue")
```

In Fig. 12.6, we see well that we now have a model that fits the fritillary counts, but that the fit comes at the expense of much less precision. Indeed, the Bayesian credible intervals are huge; much larger at any rate than under the two simpler models. This uncertainty around the estimates is a direct consequence of the introduction of the two sets of random effects. Thus, a refined analysis might try to get rid of one or both of them by introducing covariates that are informative about that variation in abundance or detection.



**FIGURE 12.6** Abundance and detection of silver-washed fritillary in the Swiss biodiversity monitoring program under a binomial mixture model with random effects in the linear predictors of both abundance and detection. Top: Mean daily abundance per transect (black: raw counts, blue: posterior median with 95% CRI, upper bound on day 5 (23) truncated). Bottom: Detection probability per individual fritillary during each of two passes on a transect.

## 12.4 SUMMARY AND OUTLOOK

Abundance  $N$  is the key numerical descriptor of a central concept in ecology, the population. Since we virtually always overlook individuals, we must usually estimate  $N$  and can not directly observe it. Classical capture–recapture methods (Williams et al., 2002; and in Chapters 6 and 10), distance sampling (Buckland et al., 2001), and spatial capture–recapture methods young (Royle and Young, 2008; Borchers and Efford, 2008; Royle et al., 2011) are well developed and can be applied to data from a single site or also to data from multiple sites. However, they are costly, in the sense that extra information in the form of individual identification or accurate distance or location measurements is needed. Frequently, ecologists are interested in abundance within a metapopulation design, where the size of a collection of local populations is needed. When replicate counts are conducted over a reasonably short period of

time, the binomial mixture model (or N-mixture model) is useful for estimation of abundance based on such relatively cheap count data. This model is an extension of the Poisson models in Chapters 3 and 4 to account for imperfect detection. The binomial mixture model is a powerful model with a big scope of application in ecology and management, such as monitoring. However, we saw that it can be difficult to find models that fit the data. Furthermore, the standard Bayesian AIC-analog, DIC, should not be used for hierarchical models such as this mixture model (Millar, 2009). Hence, in the Bayesian framework, model selection can be a challenge. A somewhat ad hoc alternative might then consist in doing model selection in the frequentist framework using AIC, for example, using functions in the new R package **unmarked** (Fiske and Chandler 2011), and then fit the best model in the Bayesian framework.

Recently, an exciting generalization of the binomial mixture model to fully open metapopulation designs has been developed by Dail and Madsen (2011); see Chandler and King (2011) for an application. This model describes the openness of local populations between successive sample periods as a function of parameters for local survival and recruitment and hence, achieves two things at the same time: providing a framework of estimating abundance, corrected for detection, without any period of closure and estimating two key parameters of population dynamics from comparatively “cheap” data. This model is an important conceptual advance for attempts at making inferences about population abundance from counts of unmarked individuals. This opens up exciting possibilities for the study of spatial population dynamics. Unfortunately, the model has so far resisted to all attempts at fitting it in WinBUGS (D. Dail, R. Chandler, A. Royle, pers. comm.), but it can be fitted using maximum likelihood in the R package **unmarked** (Fiske and Chandler, 2011).

All previous applications of open-population binomial mixture models have been more or less naive in terms of the modeled biological process. That is, explicit population dynamics models such as the Ricker model for density-dependence or a dynamics description in terms of survival and recruitment processes await to be couched within the framework of binomial mixture models. Such an integration of large-scale population dynamics modeling within an estimation framework for the latent states appears to have much promise for population ecology (Buckland et al., 2007; Hooten et al., 2007; see also Pagel and Schurr, 2011). It would open up the avenue towards the study of spatial population dynamics.

Abundance is the key state variable in ecology, so when possible we would always try to model abundance rather than simply the occurrence (distribution, “presence/absence”) of an organism in a metapopulation. However, there may be doubts about the validity of the closure assumption

(i.e., whether  $N_{ij}$  remains constant over replicates  $j$ ). In this case, it may be adequate to reduce count data to detection/nondetection data and use another variant of a hierarchical metapopulation model called a site-occupancy model. This is the topic of the final main chapter in this book.

## 12.5 EXERCISES

---

1. With hierarchical models such as the binomial mixture model, we have several kinds of covariates: here, we have covariates that vary among sites (site covariates) and those that vary among individual surveys (sampling covariates). It is important in practice to know how to fit both kinds. Invent a sampling covariate in the example of [Section 12.2.2](#) and fit it also to see how this works.
2. In the fritillary data, fit a simpler binomial mixture model than the one in [Section 12.3.3](#) with detection random effects specific to day and site (i.e., drop the index  $j$  in the  $\delta_{i,j,k}$ ). See whether that model also fits.
3. In the fritillary data, fit a more complex binomial mixture model by introducing (in addition to the random site-day-rep effect) a random site effect in the linear predictor for detection in the model in [Section 12.3.3](#). Compare the estimates under the model in [Section 12.3.3](#) and those in exercises 2 and 3. Explain.

## 13

# Estimation of Occupancy and Species Distributions from Detection/Nondetection Data in Metapopulation Designs Using Site-Occupancy Models

## OUTLINE

13.1 Introduction	414
13.2 What Happens When $p < 1$ and Constant and $p$ is Not Accounted for in a Species Distribution Model?	419
13.3 Generation and Analysis of Simulated Data for Single-Season Occupancy	420
13.3.1 The Simplest Possible Site-Occupancy Model	420
13.3.2 Site-Occupancy Models with Covariates	422
13.4 Analysis of Real Data Set: Single-Season Occupancy Model	427
13.5 Dynamic (Multiseason) Site-Occupancy Models	436
13.5.1 Generation and Analysis of Simulated Data	439
13.5.2 Dynamic Occupancy Modeling in a Real Data Set	445
13.6 Multistate Occupancy Models	450
13.7 Summary and Outlook	459
13.8 Exercises	460

## 13.1 INTRODUCTION

In much of ecology, abundance ( $N$ ) is the most interesting state variable when analyzing a population. Abundance is usually estimated from capture–recapture data or counts using the methods in Chapters 6, 10 or 12, or else strong assumptions are made about the count-abundance relationship. However, sometimes we do not have counts but only less information-rich data of the detection/nondetection kind (also misleadingly called presence/absence data). These are binary data indicating whether a species is detected (1) or not (0) at a site. We may then want to characterize one or several sites using occupancy: the probability that a site is occupied, that is, that local abundance is greater than zero. Often, occupancy is not of direct interest and merely a proxy for abundance, in which one is really interested. Indeed, it is often hard to think about occupancy separately from the abundance at the occupied sites.

However, there are also important fields in ecology that do focus on occupancy rather than abundance. Outstanding examples include meta-population ecology (Hanski, 1994, 1998), niche and species distribution (Guisan and Thuiller, 2005), and disease modeling (Thompson, 2007; McClintock et al., 2010). In addition, there is a sense in which, at a small spatial scale, occupancy and abundance coincide; when a site is chosen so small that at most one individual or pair can occupy it. The spotted owl data set in MacKenzie et al. (2003) and our [Section 13.5.1](#) provide examples for this. A similar example is given by Bled et al. (2011a), who studied the habitat selection of kittiwakes in breeding cliffs. Here, a potential nest site is a straightforward site definition, and it can be occupied by two birds at most.

This chapter deals with a class of hierarchical models known as “site-occupancy models”. In the statistical literature, these models are also called zero-inflated binomial models. In the context of distribution modeling in ecology, they have been introduced independently by MacKenzie et al. (2003) and Tyre et al. (2003), though they have important roots in earlier approaches as summarized in MacKenzie et al. (2006). “Site-occupancy model” is a fairly uninformative name for this extremely flexible modeling framework. We believe that this has helped to hide its usefulness for inference about any kind of occurrence (“presence/absence”) data at discrete sites. Essentially, site-occupancy models are hierarchical logistic regression models that jointly model the probability of occupancy and detection in animals or plants.

As usual, we believe that a hierarchical view of occurrence data is important to properly separate the ecological component and the observation component that combine to produce the observed data. However, this has not been a widely-held opinion in ecology so far. For instance, in

classical species distribution modeling (e.g., Guisan and Thuiller, 2005), it is typically ignored what is actually being modeled: it is *not* the distribution of a species. Rather, it is the *apparent species distribution* (unless detection probability is estimated). The apparent distribution is a function of both the true species distribution and of the detection probability of the species (Kéry and Schmidt, 2008; Kéry et al., 2010a; Kéry, 2011b).

There are three concerns when apparent instead of true distribution is modeled:

1. The extent of species distributions will be underestimated when  $p < 1$ ,
2. Estimates of covariate relationships will be biased towards zero when  $p < 1$ ,
3. Factors that affect the difficulty with which a species is found may end up in predictive models of species occurrence or may mask factors that do affect species occurrence.

The first is intuitively clear: if a species is not found at all sites where it occurs, the perceived range will be smaller than the actual range. However, the second is not so intuitive, especially perhaps, because it seems to be different from the modeling of abundance when detection probability is ignored. Yet, this effect has been demonstrated very clearly by Tyre et al. (2003) and in the next section, we conduct a little simulation to illustrate it. Finally, as an example of the third effect, assume that a species is more detectable in habitat A than in habitat B, for instance, because habitat A is more open and B is more wooded. In this case, open habitat may be identified as a factor that positively affects the occupancy probability/distribution of the species. For an example of the converse, see [Section 13.3.2](#).

As always, to account for imperfect detection, extra data about the observation process are required. This means temporally replicated “presence/absence” observations, where the pattern of detection/nondetection at a site contains the information about the observation process. We note that spatial replication at a small scale is informative about detection probability as well (Nichols et al., 1998a, 1998b; Kendall and White, 2009; Hines et al., 2010), but we focus on temporal replication here. Site-occupancy models require data collected in a metapopulation design (Royle, 2004c; Kéry and Royle, 2010), where (temporally or small-scale spatially) replicated detection/nondetection observations are available for a number of spatial replicates (for instance,  $> 20$ ). As in Chapter 12, analyzing such a data set does not mean to imply that it represents a metapopulation in the ecological sense of the term.

In the simplest case, we consider detection/nondetection observation  $y_{i,j}$  at site  $i$  during survey  $j$ :  $y_{i,j}$  takes on a value of 1 when a species is detected at site  $i$  on survey  $j$  and value of 0 when it is *not* detected.

It is useful to consider the genesis of all species distribution or metapopulation data as a combination of two processes: one (ecological) process determines whether a site is occupied or not and the other (observation) process determines whether the species is found or not, given that a site is occupied. Correspondingly, in a site-occupancy model, we formally distinguish between a first submodel for the partly observed true state (occurrence, the result of the ecological process) and second submodel for the actual observations. The actual observations result from both the particular realization of the ecological process and of the observation process.

$$\begin{aligned} z_i &\sim \text{Bernoulli}(\psi) & 1. \text{ Ecological process yields true state} \\ y_{i,j} | z_i &\sim \text{Bernoulli}(z_i p) & 2. \text{ Observation process yields observations} \end{aligned}$$

We naturally model true occurrence  $z_i$  ( $z_i = 1$ , if site  $i$  is occupied;  $z_i = 0$  if site  $i$  is not occupied) as a Bernoulli random variable governed by the parameter  $\psi$  (occupancy probability);  $\psi$  is the parameter that distribution modelers would wish they were modeling but only do so when detection is perfect or detection probability can be estimated. (Note that we denote probability of occupancy by  $\psi$  and the latent occurrence state of a site as  $z$ .) However,  $z_i$  is not what we usually get to see; instead, our actual observations,  $y_{i,j}$ , detection or not at site  $i$  during survey  $j$  (or “presence/absence” datum  $y_{i,j}$ ), are another Bernoulli random variable with a success rate that is the product of the actual occurrence at site  $i$ ,  $z_i$ , and detection probability  $p$  at site  $i$  during survey  $j$ . At a site where a study species does not occur,  $z$  equals 0, and  $y$  must be 0, unless there are false-positive errors. Conversely, at an occupied site, we have  $z = 1$ , and the species is detected with probability  $p$ . That is, in the site-occupancy model, detection probability is expressed *conditional on actual occurrence*, and the two parameters  $\psi$  and  $p$  are separately estimable if replicate visits are available. We could call this model a Bernoulli-Bernoulli mixture model. Moreover, recognizing that the modeling of the latent occurrence ( $z$ ) in the first level of the hierarchy accommodates additional zeroes in the data set (beyond those coming from the Bernoulli observation process), we see that it is also zero-inflated binomial (ZIB) model.

We have claimed that the term “presence/absence” for data  $y_{i,j}$  is misleading. The preceding equations clarify why this is so:  $y_{i,j}$  is a function of two processes, and only one of them has to do with occurrence and the other one is a nuisance process owing to the imperfect nature of the observation process. The true presence/absence data are the  $z_i$ , and they are only imperfectly observed and therefore latent:  $z = 1$  can be observed as  $y = 0$  or as  $y = 1$ . Site-occupancy models allow one to make a formal distinction between the two latter cases.

Two important assumptions of the model are closure and lack of false-positive errors. Closure in the context of the site-occupancy model means

that over the duration of surveys, the occurrence state of a site must not change. Each site is either occupied or it is not, but there is no extinction or colonization. This sounds like a rather strong assumption; however, it is not always that problematic. Lack of closure is akin to temporary emigration (see Chapter 9), so if temporary emigration is random, it will be confounded with detection probability. This means that temporary (but not permanent) absence of a species from a site will be one component of imperfect detection. Consequently, the estimate of the occupancy parameter will describe the proportion of sites ever occupied or used during the study period, rather than of sites that are permanently occupied, as it would in the absence of temporary emigration. If there is colonization/extinction, for instance when surveys are spread over several years, we could simply model occupancy separately for each period of closure, as we did for the open-population binomial mixture model in Section 12.3. Alternatively, we can use the dynamic occupancy model described in Section 13.5, which expresses changes in occurrence over multiple “seasons” as a function of colonization and extinction.

Absence of false positives means that no other species must be mistakenly identified as our focal species, or more generally, we must be sure that a 1 really means that our focal species was present. False positives can seriously bias occupancy estimates (Royle and Link, 2006); hence, they should be avoided for instance by good training of field personnel or by discarding doubtful records. If we discard doubtful sightings that in reality refer to our focal species, we simply lower detection probability but do not incur biased estimators. However, our models are able to deal with imperfect detection very well. When different kinds of occupancy data are available and false positives can be excluded for at least one of them, multistate occupancy models (see Section 13.6) can be used to account for both false negatives and for false positives (Miller et al., 2011).

One way to look at site-occupancy models is as a hierarchical, coupled logistic regression. One logistic regression describes true occurrence, and the other describes detection, given that the species occurs. Remember that conventional methods for distribution modeling (GLM, GAM, boosted regression trees: Elith et al., 2008; Maxent: Phillips and Dudik, 2008) would pool the temporal replicates  $j$  and model the maximal observation, that is, site  $i$  will get a value of 1 if the species was ever detected there. Those approaches discard the information available about the observation process and thus in principle cannot model true, but only apparent species distributions (Kéry et al., 2010a). In contrast, site-occupancy models exploit all the available information about both ecological and observation process contained in detection/nondetection data.

The two Bernoulli distributions above describe the simplest possible site-occupancy model, where both occupancy ( $\psi$ ) and detection probability ( $p$ )

are constant (see [Section 13.3.1](#)). This simple model can be extended in many ways. Most importantly, we need to be able to model the effects of measured covariates on one or both parameter(s). Both the ecological and the observation processes represent a logistic regression (with an intercept only so far), so it is natural to include covariate effects via a logit link function. Hence, we can add statements of the following kind to the model description

$$\text{logit}(\psi_i) = \alpha + \beta * x_i.$$

Here,  $x_i$  is the value of some occurrence-relevant covariate measured at site  $i$ , and  $\alpha$  and  $\beta$  are the intercept and slope parameters of this logit-linear regression. We can do the same for the observation model, where we distinguish between “site covariates” and “sampling covariates”. Site covariates vary among sites only and are constant across repeated surveys to a site, that is, they will be indexed by  $i$  only. In contrast, survey covariates vary by site *and* by survey; hence, they will be indexed by  $i$  and  $j$ . This is a minor distinction but in practice, the modeling of sampling covariates requires a little more book-keeping effort. Explicitly couching site-occupancy models within the GLM framework makes it clear that other GLM extensions might be applied, too. For instance, overdispersion in detection probability could be modeled by the introduction of random site effects (Royle, 2006). Of course, we could model the effects of many explanatory variables, of polynomial terms, or of splines (Gimenez et al., 2006a, b; Collier et al., 2011).

In [Section 13.2](#), we conduct a simulation to understand what happens to the estimates of regression coefficients in conventional species distribution models when detection probability is not perfect. In [Section 13.3](#), we analyze simulated data sets and in [Section 13.4](#) a real data set using single-season site-occupancy models. In [Section 13.5](#), we extend the model to multiple “seasons” and thus arrive at an extended metapopulation model. In [Section 13.6](#), we extend the single-season model to multiple states of occurrence, which in our example are owl territories occupied with or without reproduction.

We emphasize that we will not conduct any goodness-of-fit assessments based on posterior predictive checks in this chapter. The reason for this is that with a binary response, the deviance or other discrepancy measures based directly on the response are uninformative about the fit of a model (McCullagh and Nelder, 1989). Kéry (2010) erroneously showed such posterior predictive checks for site-occupancy models. These checks are meaningless because regardless of the model structure, they will always and thus sometimes spuriously indicate a fitting model. To do a goodness-of-fit test, the binary responses have to be aggregated.

## 13.2 WHAT HAPPENS WHEN $p < 1$ AND CONSTANT AND $p$ IS NOT ACCOUNTED FOR IN A SPECIES DISTRIBUTION MODEL?

We use simulation to understand what happens when there is a constant degree of imperfect detection and this is not accounted for in an analysis. We simulate 100,000 data sets from 250 sites, with a constant  $p < 1$  (here,  $p = 0.60$ ), and analyze them with a conventional species distribution model (here, a nonhierarchical logistic regression). We have a single explanatory variable (think of it as a habitat or environmental covariate) that links the habitat to occurrence probability on the logit-linear scale with intercept  $-3$  and slope  $1$ . (You may want to change `nreps` in the code to 1000.)

```

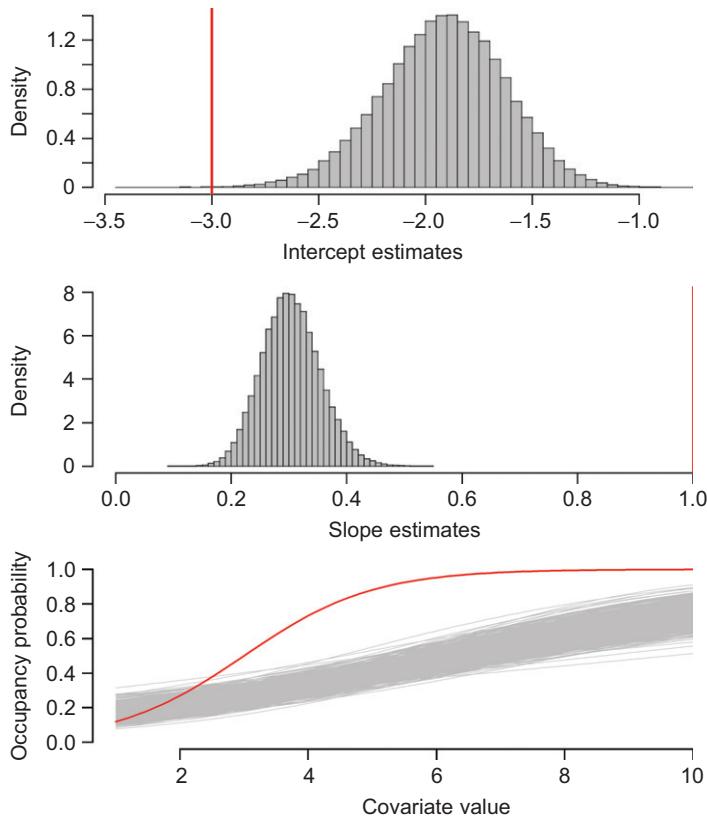
nreps <- 10^5
estimates <- array(NA, dim=c(nreps, 2))      # No. replicates
                                                    # Array to contain the
                                                    # estimates
R <- 250                                         # No. sites

for (i in 1:nreps) {
  cat(i, "\n"); flush.console()
  x <- runif(R, 0, 10) # choose covariate values
  state<-rbinom(n=R, size=1, prob=plogis(-3 + 1 * x)) # Occ. state
  obs <- rbinom(n=R, size=1, prob=0.6) * state          # Observations
  fm <- glm(obs~x, family=binomial)
  estimates[i,] <- fm$coef
}

par(mfrow=c(3, 1))
hist(estimates[,1], col = "gray", nclass=50, main="",
     xlab = "Intercept estimates", las=1, ylab = "", freq=FALSE)
abline(v = -3, col = "red", lwd=3)                 # Truth
hist(estimates[,2], col = "gray", nclass=50, main="",
     xlab = "Slope estimates", xlim=c(0,1), las=1, ylab = "", freq=FALSE)
abline(v = 1, col = "red", lwd=3)                   # Truth
plot(1:10, plogis(estimates[1,1] + estimates[1,2] * (1:10)), col =
     "gray", lwd=1, ylab = "Occupancy probability", xlab = "Covariate
     value", type = "l", ylim=c(0, 1), frame.plot = FALSE, las=1)
samp <- sample(1:nreps, 1000)
for (i in samp){
  lines(1:10, plogis(estimates[i,1] + estimates[i,2] * (1:10)),
        col = "gray", lwd=1, type = "l")
}
lines(1:10, plogis(-3 + 1 * (1:10)), col = "red", lwd=3, type = "l")

```

When failing to account for a constant nondetection error, slope estimates of a covariate are biased towards zero (Fig. 13.1, middle panel). The intercept (Fig. 13.1, top panel) is not necessarily estimated too low; rather, here, it is overestimated. However, the combined effect is such



**FIGURE 13.1** Effect of imperfect detection on a conventional species distribution model: slope estimates become biased low with imperfect detection even if detection probability is constant (here, 0.60). In the bottom panel, the red lines show the truth and the gray lines show a random sample of 1000 estimated regression lines: the extent of the distribution is always underestimated. See also Tyre et al. (2003).

that the total extent of a distribution is underestimated. The latter is represented by the area under the red curve in the bottom panel. The area under the gray curves (the estimated distribution) is always less than the area under the red curve (true distribution).

### **13.3 GENERATION AND ANALYSIS OF SIMULATED DATA FOR SINGLE-SEASON OCCUPANCY**

#### **13.3.1 The Simplest Possible Site-Occupancy Model**

To fully grasp how the site-occupancy model “works”, we first look at the simplest possible case: both the ecological and the observation process

are described by an intercept only. To generate detection/nondetection data  $y_{ij}$  under this Null model for  $R = 200$  spatial replicates (sites) and  $T = 3$  temporal replicates, we simply do this.

```
# Select sample sizes (spatial and temporal replication)
R <- 200
T <- 3

# Determine process parameters
psi <- 0.8      # Occupancy probability
p <- 0.5        # Detection probability

# Create structure to contain counts
y <- matrix(NA, nrow=R, ncol=T)

# Ecological process: Sample true occurrence (z, yes/no) from a
# Bernoulli (occurrence probability=psi)
z <- rbinom(n=R, size=1, prob=psi) # Latent occurrence state

# Observation process: Sample detection/nondetection observations
# from a Bernoulli (with p) if z=1
for (j in 1:T) {
  y[,j] <- rbinom(n=R, size=1, prob=z * p)
}

# Look at truth and at our imperfect observations
sum(z)           # Realized occupancy among 200 surveyed sites
[1] 169
sum(apply(y, 1, max)) # Observed occupancy
[1] 151
```

Note that in the simulation of the observation process, we have multiplied the Bernoulli draw with  $z$ . This means that the result will be zero whenever  $z=0$ , that is, whenever the species does not occur. Next, we analyze this data set.

```
# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
psi ~ dunif(0, 1)
p ~ dunif(0, 1)

# Likelihood
# Ecological model for true occurrence
for (i in 1:R) {
  z[i] ~ dbern(psi)
  p.eff[i] <- z[i] * p

  # Observation model for replicated detection/nondetection
  # observations
  for (j in 1:T) {
    y[i,j] ~ dbern(p.eff[i])
  }
}
```

```

        } #j
    } #i

# Derived quantities
occ.fs <- sum(z[])
# Number of occupied sites among the 200
}
",fill = TRUE)
sink()

# Bundle data
win.data <- list(y=y, R=nrow(y), T=ncol(y))

# Initial values
zst <- apply(y, 1, max) # Observed occurrence as starting values for z
inits <- function() list(z=zst)

# Parameters monitored
params <- c("psi", "p", "occ.fs")

# MCMC settings
ni <- 1200
nt <- 2
nb <- 200
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
bugs.dir, working.directory=getwd())

# Summarize posteriors
print(out, dig=2)
[...]
      mean     sd   2.5%   25%   50%   75%  97.5% Rhat n.eff
psi      0.89  0.04   0.80   0.86   0.89   0.92   0.97  1.01  340
p       0.47  0.03   0.41   0.45   0.47   0.49   0.53  1.00  870
occ.fs  178.29 7.59 165.00 173.00 178.00 183.00 195.00 1.01  510
deviance 739.37 28.50 686.50 719.00 738.20 758.05 798.75 1.01  470
[...]

```

This looks good. You will note quite a bit of sampling variability in this system. The estimates may be fairly different among repeated generations of the data set or among the replicate data sets of different people. This basic model is a good starting point for running simulation exercises to find out about how good inferences can be in marginal data situations; see exercises and Guillera-Arroita et al. (2010).

### 13.3.2 Site-Occupancy Models with Covariates

Next, we look into the case where covariates affect the ecological and the observation process. We model covariate effects on a parameter  $\theta$  through the canonical GLM link function, the  $\text{logit} = \log(\theta/(1-\theta))$ . As in the previous chapter, we will look at a worst-case scenario for a species

distribution model, where opposing effects of a single covariate on the two processes generating the observed data effectively cancel each other out in the observations. The result will be that in a conventional species distribution model, the effect of this covariate on the species distribution will not be identified correctly. We next define a function that creates species distribution data (detection/nondetection data) for us.

```
# Define function for generating species distribution data
data.fn <- function(R = 200, T = 3, xmin = -1, xmax = 1, alpha.psi = -1,
beta.psi = 3, alpha.p = 1, beta.p = -3) {
  y <- array(dim = c(R, T)) # Array for counts

  # Ecological process
  # Covariate values
  X <- sort(runif(n = R, min = xmin, max = xmax))

  # Relationship expected occurrence - covariate
  psi <- plogis(alpha.psi + beta.psi * X) # Apply inverse logit

  # Add Bernoulli noise: draw occurrence indicator z from
  # Bernoulli(psi)
  z <- rbinom(n = R, size = 1, prob = psi)
  occ.fs <- sum(z) # Finite-sample occupancy (see
  Royle and Kéry, 2007)

  # Observation process
  # Relationship detection prob - covariate
  p <- plogis(alpha.p + beta.p * X)

  # Make a 'census'
  p.eff <- z * p
  for (i in 1:T) {
    y[, i] <- rbinom(n = R, size = 1, prob = p.eff)
  }

  # Naïve regression
  naive.pred <- plogis(predict(glm(apply(y, 1, max) ~ X + I(X^2),
family = binomial)))

  # Plot features of the simulated system
  par(mfrow = c(2, 2))
  plot(X, psi, main = "Expected occurrence", xlab = "Covariate",
       ylab = "Occupancy probability", las = 1, type = "l", col = "red",
       lwd = 3, frame.plot = FALSE)
  plot(X, z, main = "Realised (true) occurrence", xlab = "Covariate",
       ylab = "Occurrence", las = 1, frame.plot = FALSE, col = "red")
  plot(X, p, ylim = c(0, 1), main = "Detection probability",
       xlab = "Covariate", ylab = "p", type = "l", lwd = 3, col = "red",
       las = 1, frame.plot = FALSE)
  plot(X, naive.pred, main = "Detection/nondetection observations \n and conventional SDM", xlab = "Covariate", ylab = "Apparent
occupancy", ylim = c(min(y), max(y)), type = "l", lwd = 3, lty = 2,
col = "blue", las = 1, frame.plot = FALSE)
  points(rep(X, T), y)
```

```
# Return stuff
return(list(R=R, T=T, X=X, alpha.psi=alpha.psi, beta.psi=
beta.psi, alpha.p=alpha.p, beta.p=beta.p, psi=psi, z=z,
occ.fs=occ.fs, p=p, y=y))
}
```

We obtain one realization from the stochastic system just defined and conduct a conventional species distribution model (Fig. 13.2):

```
sodata <- data.fn()
str(sodata)                      # Look at data
summary(glm(apply(y, 1, max) ~ X + I(X^2), family=binomial,
data=sodata))

Call:
glm(formula = apply(y, 1, max) ~ X + I(X^2), family = binomial,
     data = sodata)

Deviance Residuals:
    Min          1Q      Median          3Q      Max
 -1.10984   -0.83363   -0.28985   -0.04219    2.45653

Coefficients:
            Estimate Std. Error   z value Pr(>|z|)
(Intercept) -1.0439    0.2624 -3.978   6.95e-05 ***
X             3.3989    0.8348  4.072   4.67e-05 ***
I(X^2)       -3.2680    1.1757 -2.780   0.00544 **
---
Signif. codes: 0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 213.27 on 199 degrees of freedom
Residual deviance: 170.95 on 197 degrees of freedom
AIC: 176.95

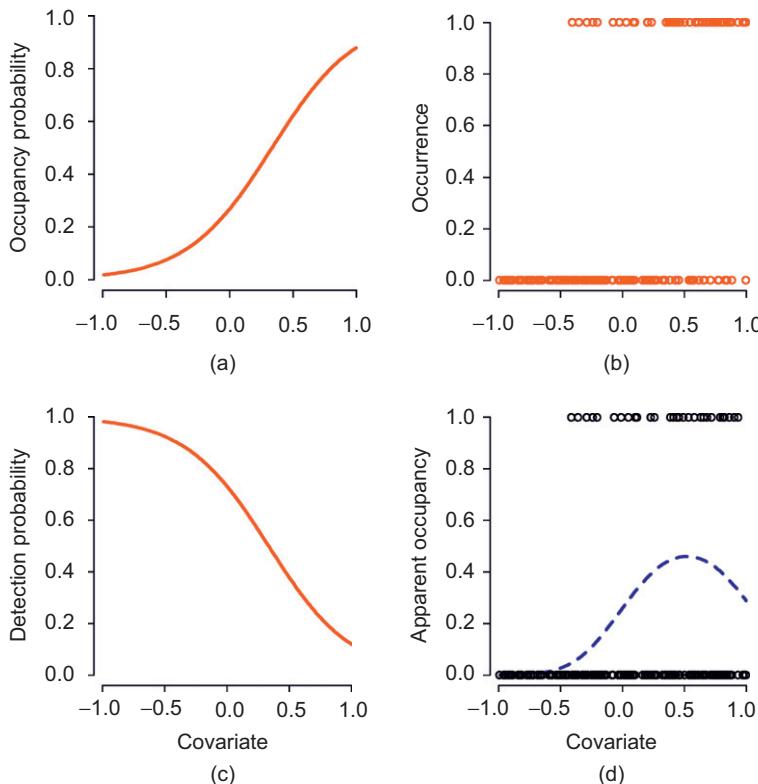
Number of Fisher Scoring iterations: 6
```

Hence, in this simulated data set and with a conventional species distribution model, we identify an optimum value of the covariate for the occupancy probability of the study species (see blue curve in bottom right panel of Fig. 13.2). Let us see what a site-occupancy model can do.

```
# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
alpha.occ ~ dunif(-10, 10)
beta.occ ~ dunif(-10, 10)
alpha.p ~ dunif(-10, 10)
beta.p ~ dunif(-10, 10)

# Likelihood
for (i in 1:R) {
```



**FIGURE 13.2** Features of the simulated data set, and truth behind it, and inference about the system based on a conventional species distribution model (blue line in bottom right panel). The truth is shown in red and observed data in black. (a) Occupancy probability, (b) realized (true) occurrence, (c) detection probability, and (d) detection/nondetection (“presence/absence”) observations and estimated occupancy probability under a conventional species distribution model.

```
# True state model for the partially observed true state
z[i] ~ dbern(psi[i])           # True occupancy z at site i
logit(psi[i]) <- alpha.occ + beta.occ * X[i]
for (j in 1:T) {
  # Observation model for the actual observations
  y[i,j] ~ dbern(p.eff[i,j]) # Detection-nondetection at i and j
  p.eff[i,j] <- z[i] * p[i,j]
  logit(p[i,j]) <- alpha.p + beta.p * X[i]
  } #j
} #i

# Derived quantities
occ.fs <- sum(z[]) # Number of occupied sites among those studied
}
", fill = TRUE)
sink()
```

```

# Bundle data
win.data <- list(y = sodata$y, X = sodata$X, R = nrow(sodata$y),
                  T = ncol(sodata$y))

# Initial values
zst <- apply(sodata$y, 1, max) # Good inits for latent states essential
inits <- function(){list(z=zst, alpha.occ=runif(1, -3, 3),
                         beta.occ=runif(1, -3, 3), alpha.p=runif(1, -3, 3), beta.p=runif
                         (1, -3, 3))}

# Parameters monitored
params <- c("alpha.occ", "beta.occ", "alpha.p", "beta.p", "occ.fs")

# MCMC settings
ni <- 10000
nt <- 8
nb <- 2000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
            n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
            bugs.dir, working.directory=getwd())

```

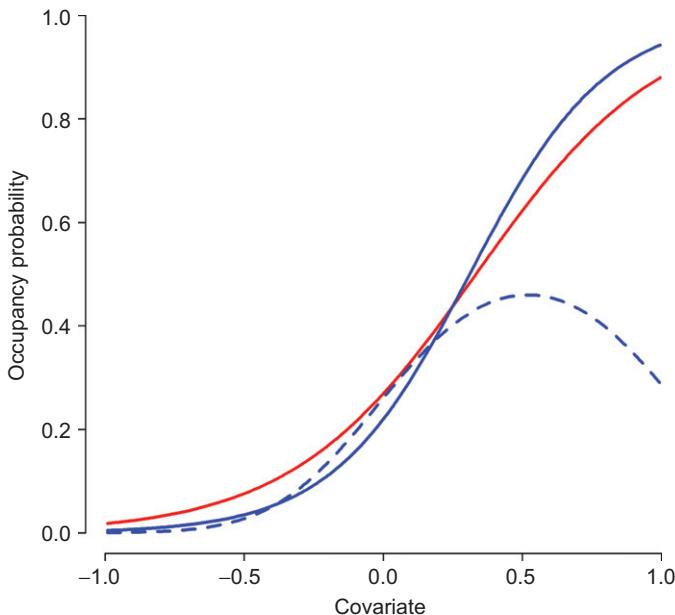
We compare the known truth in the data-generating mechanism with our estimates of truth under the site-occupancy species distribution model. We find that the model does a decent job at recovering the parameters for the habitat relationships of the probability of occupancy (`alpha.occ` and `beta.occ`) and of detection (`alpha.p` and `beta.p`), but that the estimates are much more precise for the relationship with detection. This makes sense because there is more data ( $n = 600$  instead of  $n = 200$ ) from which to estimate those regression parameters. A total of 59 sites were occupied in our simulated data set, and at 45 of those, the study species was discovered. Our model estimated 67 occurrences (95% CRI 57–78). This number, finite-sample occurrence, is not a function of population occupancy probability, but of the latent occurrence states  $z$ , which we can easily estimate in an MCMC-based analysis (Royle and Kéry, 2007).

```

TRUTH <- c(sodata$alpha.psi, sodata$beta.psi, sodata$alpha.p, sodata
            $beta.p, sum(sodata$z))
print(cbind(TRUTH, out$summary[1:5, c(1,2,3,7)]), dig=3)
      TRUTH    mean     sd  2.5%   97.5%
alpha.occ     -1 -1.269  0.274 -1.81  -0.738
beta.occ      3  4.084  0.854  2.58   5.939
alpha.p       1  0.925  0.330  0.28   1.584
beta.p       -3 -2.942  0.546 -4.00  -1.865
occ.fs       59 67.335  5.291 57.00  78.000
sum(apply(sodata$y, 1, sum) > 0) # Apparent number of occupied sites
[1] 45

```

We graphically compare the conclusions from the two species distribution models (Fig. 13.3). We see again that the conventional approach,



**FIGURE 13.3** Comparison of true and estimated relationship between occupancy probability and an environmental covariate under a site-occupancy model (solid blue) and under the conventional approach that ignores detection probability (dashed blue). Truth is shown in red.

which ignores the effects of the observation process in the generation of detection/nondetection data, models apparent rather than true species distributions only (Kéry, 2011b).

```
naive.pred <- plogis(predict(glm(apply(sodata$y, 1, max) ~ X + I(X^2),
  family=binomial, data=sodata)))
lin.pred2 <- out$mean$alpha.occ + out$mean$beta.occ * sodata$X
plot(sodata$X, sodata$psi, ylim=c(0, 1), main = "", ylab = "Occupancy
  probability", xlab = "Covariate", type = "l", lwd = 3, col = "red",
  las = 1, frame.plot = FALSE)
lines(sodata$X, naive.pred, ylim=c(0 ,1), type = "l", lty = 2, lwd = 3,
  col = "blue")
lines(sodata$X, plogis(lin.pred2), ylim=c(0, 1), type = "l", lty = 1,
  lwd = 2, col = "blue")
```

## 13.4 ANALYSIS OF REAL DATA SET: SINGLE-SEASON OCCUPANCY MODEL

We will next analyze a small, but typical real-world occurrence data set: surveys to breeding sites of the endangered beetle *Rosalia alpina* (Fig. 13.4; see also the cover of Kéry, 2010) during a single flight period (July–August



**FIGURE 13.4** The remarkable “blue bug”, the cerambycid beetle *Rosalia alpina*, Switzerland, 2009 (Photograph by T. Marent).

2009). In Switzerland, this striking blue bug lays its eggs into the wood of dead beech trees *Fagus sylvatica*, preferentially in tall and old logs, but unfortunately also in piles of firewood stocked in the forest only temporarily. Larvae develop over 3–4 years; hence, eggs laid in firewood are normally doomed. Nevertheless, checking firewood piles in forests is an efficient search strategy for this rare and elusive beetle. In 2009, one of us (MK) surveyed one of the few Swiss areas where the species is known to occur, the hills around Movelier in the Swiss Jura mountains.

The complete data set (“bluebug.txt”) contains replicated counts at a total of 27 sites (woodpiles) in the Movelier region in 2009. There were up to six replicate counts at each woodpile; the count result of which is called `detX`. Woodpiles were either at the forest edge or more in the interior of a forest (`covariate forest_edge`), and individual visits took place at varying dates (`covariate dateX`) and times of day (hours in the afternoon, covariates `hx`).

A summary of these data is shown in Table 13.1. We see that *Rosalia* was detected at 10 of 27 woodpiles and from 1 to 5 times. Clearly, detection probability at an occupied woodpile is not perfect; for instance, the woodpile in row 10 was surveyed six times and *Rosalia* was seen only once. It is natural to wonder whether other woodpiles might have been occupied but *Rosalia* was simply missed. Another question might be to ask how many times a woodpile might have to be checked in order to detect *Rosalia* at least once when it occurs. And finally, we may wonder

**TABLE 13.1** A Summary of the Blue Bug Data Set (*bluebug.txt*) That Keeps Track Only of Detections and Nondetections.

0	1	1	1	1	1	5
1	1	1	1	1	-	5
1	0	1	0	0	1	3
1	0	0	0	1	1	3
1	1	-	-	-	-	2
1	-	-	-	-	-	1
0	0	0	0	1	0	1
1	-	-	-	-	-	1
1	-	-	-	-	-	1
1	0	0	0	0	0	1
0	0	0	0	0	-	0
0	0	0	0	0	-	0
0	0	0	0	0	-	0
0	-	-	-	-	-	0
0	-	-	-	-	-	0
0	-	-	-	-	-	0
0	-	-	-	-	-	0
0	0	-	-	-	-	0
0	0	-	-	-	-	0
0	0	-	-	-	-	0
0	0	0	-	-	-	0
0	-	-	-	-	-	0
0	-	-	-	-	-	0
0	-	-	-	-	-	0

Note: Rows denote woodpiles and columns, except for the right-most column, denote survey occasions. The total number of surveys with detections is shown in the right-most column. Surveys with *Rosalia* detections are shown in buff color, those without *Rosalia* detections in yellow, and missing values shown as dashes. For pure convenience, sites have been ordered by decreasing number of surveys with detections.

whether the location of a woodpile, at the forest edge or in the interior, may affect the probability of it being occupied, and similarly, whether there were relationships between detection probability and the date and time of day, respectively, at which a survey took place, or whether *Rosalia* was detected before (behavioral effect, see Section 6.2.3.). We will answer these questions with a site-occupancy species distribution model now.

```
# Read in the data
data <- read.table("bluebug.txt", header = TRUE)

# Collect the data into suitable structures
y <- as.matrix(data[,4:9])           # as.matrix essential for WinBUGS
y[y>1] <- 1                         # Reduce counts to 0/1
edge <- data$forest_edge
dates <- as.matrix(data[,10:15])
hours <- as.matrix(data[,16:21])

# Standardize covariates
mean.date <- mean(dates, na.rm = TRUE)
sd.date <- sd(dates[!is.na(dates)])
DATES <- (dates-mean.date)/sd.date      # Standardise date
DATES[is.na(DATES)] <- 0                # Impute zeroes (means)

mean.hour <- mean(hours, na.rm = TRUE)
sd.hour <- sd(hours[!is.na(hours)])
HOURS <- (hours-mean.hour)/sd.hour     # Standardise hour
HOURS[is.na(HOURS)] <- 0                # Impute zeroes (means)
```

In the BUGS code below, we “stabilize” the logit to avoid numerical under- or overflow by truncating values more extreme than (-999, 999) on the logit scale. This should hardly affect the inference because this restricts the value of the linear predictor to the range (plogis(-999), plogis(999)).

```
# Specify model in BUGS language
sink("model.txt")
cat("
model {

# Priors
alpha.psi ~ dnorm(0, 0.01)
beta.psi ~ dnorm(0, 0.01)
alpha.p ~ dnorm(0, 0.01)
beta1.p ~ dnorm(0, 0.01)
beta2.p ~ dnorm(0, 0.01)
beta3.p ~ dnorm(0, 0.01)
beta4.p ~ dnorm(0, 0.01)

# Likelihood
# Ecological model for the partially observed true state
for (i in 1:R) {
  z[i] ~ dbern(psi[i]) # True occurrence z at site i
  psi[i] <- 1 / (1 + exp(-lpsi.lim[i]))
  lpsi.lim[i] <- min(999, max(-999, lpsi[i]))
  lpsi[i] <- alpha.psi + beta.psi * edge[i]
```

```

# Observation model for the observations
for (j in 1:T) {
  y[i,j] ~ dbern(mu.p[i,j]) # Detection-nondetection at i and j
  mu.p[i,j] <- z[i] * p[i,j]
  p[i,j] <- 1 / (1 + exp(-lp.lim[i,j]))
  lp.lim[i,j] <- min(999, max(-999, lp[i,j]))
  lp[i,j] <- alpha.p + beta1.p * DATES[i,j] + beta2.p *
    pow(DATES[i,j], 2) + beta3.p * HOURS[i,j] + beta4.p *
    pow(HOURS[i,j], 2)
  } #j
} #i

# Derived quantities
occ.fs <- sum(z[]) # Number of occupied sites
mean.p <- exp(alpha.p) / (1 + exp(alpha.p)) # Average detection
}
",fill=TRUE)
sink()

# Bundle data
win.data <- list(y=y, R=nrow(y), T=ncol(y), edge=edge, DATES =
  DATES, HOURS = HOURS)

# Initial values
zst <- apply(y, 1, max, na.rm=TRUE) # Good starting values crucial
inits <- function(){list(z=zst, alpha.psi=runif(1, -3, 3), alpha.p=
  runif(1, -3, 3))}

# Parameters monitored
params <- c("alpha.psi", "beta.psi", "mean.p", "occ.fs", "alpha.p",
  "beta1.p", "beta2.p", "beta3.p", "beta4.p")

# MCMC settings
ni <- 30000
nt <- 10
nb <- 20000
nc <- 3

# Call WinBUGS from R (BRT < 1 min)
out <- bugs(win.data, inits, params, "model.txt", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug=TRUE, bugs.directory =
  bugs.dir, working.directory=getwd())

```

We inspect the estimates and then illustrate.

```

# Summarize posteriors
print(out, dig=2)
      mean     sd   2.5%   25%   50%   75%  97.5%   Rhat   n.eff
alpha.psi  5.83   5.26  -0.10   1.73   4.26   8.67  17.98   1.10      46
beta.psi  -6.61   5.26 -18.83  -9.38  -5.13  -2.60  -0.44   1.10      48
mean.p    0.56   0.15   0.27   0.46   0.56   0.67   0.85   1.01     200
occ.fs   17.02   2.38  11.00  16.00  17.00  18.00  21.00   1.01     220
alpha.p   0.29   0.66  -0.97  -0.15   0.26   0.72   1.71   1.01     160
beta1.p   0.34   0.40  -0.42   0.06   0.33   0.60   1.13   1.00     2400
beta2.p   0.21   0.47  -0.71  -0.10   0.19   0.51   1.17   1.01     230
beta3.p  -0.48   0.42 -1.37  -0.75  -0.46  -0.20   0.31   1.01     330
beta4.p  -0.59   0.32 -1.28  -0.79  -0.57  -0.37   0.00   1.00     1600

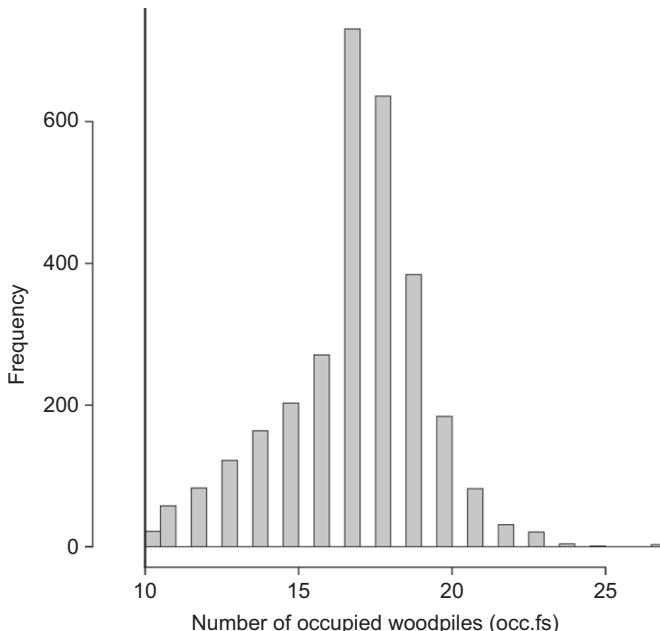
```

We note that convergence for the occupancy parameters could be better ( $Rhat = 1.10$ ). We also note (not shown) that parameter estimates are quite sensitive to the priors chosen in the model. This is not quite unexpected, given the small size of the data set. Thus, we should state our inferences with caution.

Earlier on, we asked a series of questions that we wanted to answer with the site-occupancy model. The first was “How many woodpiles were likely occupied by *Rosalia alpina*, given the detection probability estimated?”. We find the answer in the tabular summary of the estimates above, it is 17.02 (95% CRI 11–21). Since this is a key quantity in our analysis, we want to visualize its entire posterior distribution (Fig. 13.5).

```
# Posterior distribution of the number of occupied woodpiles in actual
# sample
hist(out$sims.list$occ.fs, nclass = 30, col = "gray", main = "", xlab =
  "Number of occupied woodpiles (occ.fs)", xlim = c(9, 27))
abline(v = 10, lwd = 2) # The observed number
```

The second question of interest was “Given that we may overlook the species at occupied woodpiles, how many times must we survey a



**FIGURE 13.5** Posterior distribution of the number of woodpiles occupied by the cerambycid beetle *Rosalia alpina* in the Movelier region in 2009 among the 27 surveyed woodpiles. Vertical line indicates the observed number of 10.

woodpile before we can be “almost certain” to detect it at least once, when it occurs?”. We can answer this question by using a simple binomial argument put forwards in Kéry (2002) and many times elsewhere: the probability  $P^*$  to detect the species during  $n$  identical and independent surveys is  $P^* = 1 - (1 - p)^n$ , where  $p$  is the detection probability from a site-occupancy model. Since detection varies in all sorts of ways (see below), we have to decide on one “useful” value of  $p$ . We take the `mean.p` monitored in the analysis. Using the MCMC samples for that quantity, we can incorporate our uncertainty about detection probability into the answer to our question. We will compute  $P^*$  for values of  $n$  between 1 and 10 and see where it is at least 95%, which will be our definition of “almost certain”.

```
Pstar <- array(NA, dim=c(out$n.sims, 10))
x <- cbind(rep(1, 3000), rep(2, 3000), rep(3, 3000), rep(4, 3000), rep
           (5, 3000), rep(6, 3000), rep(7, 3000), rep(8, 3000), rep(9, 3000),
           rep(10, 3000))
for (i in 1:out$n.sims) {
  for (j in 1:10) {
    Pstar[i,j] <- 1 - (1 - out$sims.list$mean.p[i])^j
  } #j
} #i
boxplot(Pstar~x, col = "gray", las = 1, ylab = "Pstar", xlab = "Number of
         surveys", outline = FALSE)
abline(h = 0.95, lty = 2, lwd = 2)
```

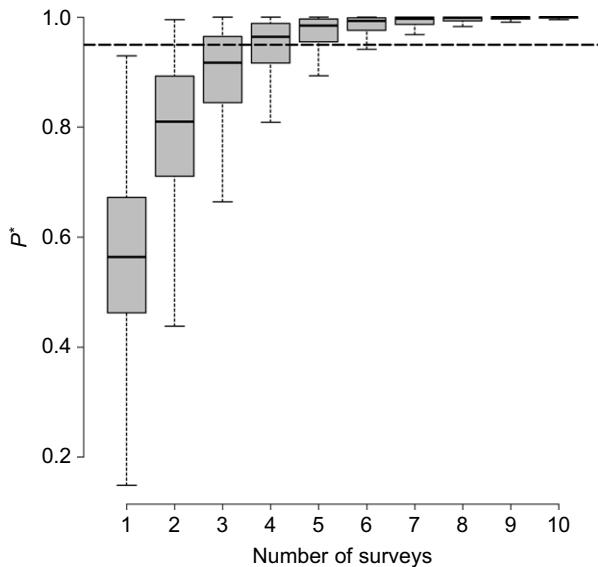
Hence, 3–4 “average” surveys were required to be almost certain to detect *Rosalia alpina* at a woodpile where it occurred (Fig. 13.6).

What about the occupancy at woodpiles at the forest edge as compared to the forest interior? Our parameter `beta.psi` represents the difference in occupancy probability, on the logit scale between woodpiles at the forest edge and those in the interior. The 95% CRI of its estimate does not cover 0; hence, we can be rather confident in that *Rosalia* was more widespread at woodpiles in the forest interior. We convert the occupancy parameters into an estimate of occupancy in both locations and plot that.

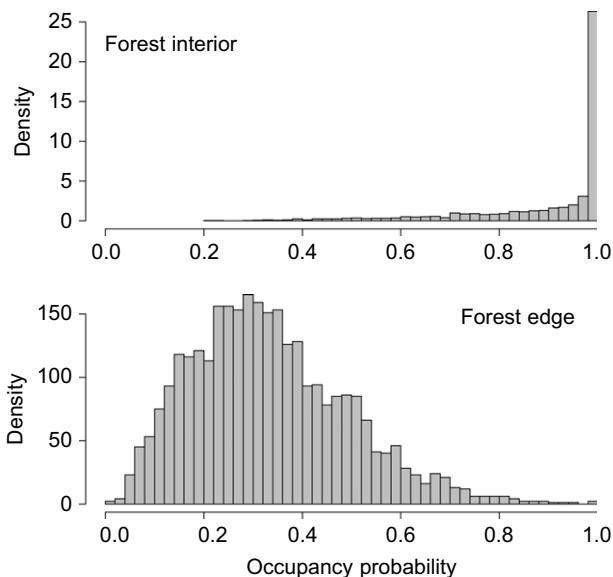
```
par(mfrow = c(2, 1))
hist(plogis(out$sims.list$alpha.psi), nclass = 40, col = "gray", main =
      "Forest interior", xlab = "Occupancy probability", xlim = c(0, 1))
hist(plogis(out$sims.list$alpha.psi + out$sims.list$beta.psi),
      nclass = 40, col = "gray", main = "Forest edge", xlab = "Occupancy
      probability", xlim = c(0, 1))
```

So, indeed, there appears to be a big effect of the location of a woodpile on the probability that it is occupied by *Rosalia alpina*: the forest interior is much preferred (Fig. 13.7).

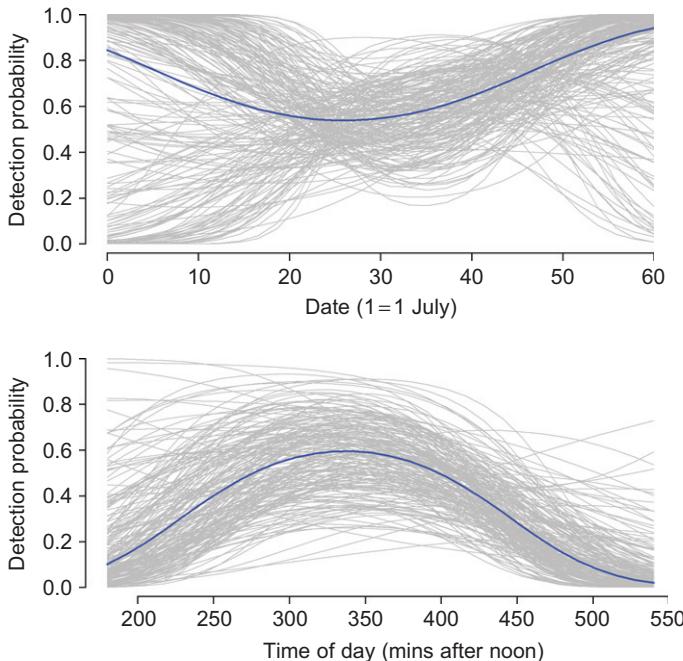
Finally, we want to answer the questions about a relationship between detection probability and date and time of day, respectively. We can see from the 95% CRI in the summary results table above that the regression



**FIGURE 13.6** The relationship between  $P^*$ , the probability to detect *Rosalia alpina* at a woodpile at least once during  $n$  surveys, and  $n$  for the blue bug data set. The dashed line indicates 95% certainty to detect the species when present.



**FIGURE 13.7** Posterior distributions of the probability of occupancy by *Rosalia alpina* for a woodpile in the forest interior (top) and at the forest edge (bottom) in Movelier, 2009.



**FIGURE 13.8** Predictions of the covariate relationships that account for estimation uncertainty. Top, effect of date; bottom, effect of time of day. Blue lines show the posterior mean, and gray lines show the relationships based on a random posterior sample of size 200 to visualize estimation uncertainty.

parameters for date, beta1.p and beta2.p, largely overlap zero but that those for time of day, beta3.p and beta4.p, do not do this so clearly (at least not beta4.p, which just about straddles 0). We will plot the predicted relationship in a figure that also shows the uncertainty in the estimates by plotting the relationships for a random MCMC sample of the regression coefficients involved in their computation (Fig. 13.8). This again suggests the absence of a date effect on detection probability (top panel); however, detection probability seems to be highest around 5–6 pm (bottom panel). These results can be interesting for designing a monitoring program for this endangered species.

```
# Predict effect of time of day with uncertainty
mcmc.sample <- out$n.sims

original.date.pred <- seq(0, 60, length.out = 30)
original.hour.pred <- seq(180, 540, length.out = 30)
date.pred <- (original.date.pred - mean.date)/sd.date
hour.pred <- (original.hour.pred - mean.hour)/sd.hour
p.pred.date <- plogis(out$mean$alpha.p + out$mean$beta1.p *
  date.pred + out$mean$beta2.p * date.pred^2 )
```

```

p.pred.hour <- plogis(out$mean$alpha.p + out$mean$beta3.p *
  hour.pred + out$mean$beta4.p * hour.pred^2)

array.p.pred.hour <- array.p.pred.date <- array(NA, dim = c(length
  (hour.pred), mcmc.sample))
for (i in 1:mcmc.sample){
  array.p.pred.date[,i] <- plogis(out$sims.list$alpha.p[i] +
    out$sims.list$beta1.p[i] * date.pred + out$sims.list$beta2.p[i] *
    date.pred^2)
  array.p.pred.hour[,i] <- plogis(out$sims.list$alpha.p[i] +
    out$sims.list$beta3.p[i] * hour.pred + out$sims.list$beta4.p[i] *
    hour.pred^2)
}

# Plot for a subsample of MCMC draws
sub.set <- sort(sample(1:mcmc.sample, size = 200))

par(mfrow=c(2, 1))
plot(original.date.pred, p.pred.date, main = "", ylab = "Detection
  probability", xlab = "Date (1 = 1 July)", ylim = c(0, 1), type = "l",
  lwd = 3, frame.plot = FALSE)
for (i in sub.set){
  lines(original.date.pred, array.p.pred.date[,i], type = "l",
    lwd = 1, col = "gray")
}
lines(original.date.pred, p.pred.date, type = "l", lwd = 3,
  col = "blue")

plot(original.hour.pred, p.pred.hour, main = "", ylab = "Detection
  probability", xlab = "Time of day (mins after noon)", ylim = c(0, 1),
  type = "l", lwd = 3, frame.plot = FALSE)
for (i in sub.set){
  lines(original.hour.pred, array.p.pred.hour[,i], type = "l",
    lwd = 1, col = "gray")
}
lines(original.hour.pred, p.pred.hour, type = "l", lwd = 3,
  col = "blue")

```

## 13.5 DYNAMIC (MULTISEASON) SITE-OCCUPANCY MODELS

So far we have been modeling detection/nondetection observations from  $R$  sites and  $J$  replicate surveys, yielding data  $y_{i,j}$  for site  $i$  and survey  $j$ . We required a so-called closed population, which in the occupancy context means that the occurrence state of site  $i$  must not change over the  $J$  replicates. The closure assumption is often a reasonable approximation for studies that are short relative to the dynamics of the system investigated. However, in other cases, closure may not hold for all replicate surveys, for instance, when animals randomly move onto and off study sites. This specific form of nonclosure is called random temporary emigration, and the models of the preceding sections may still be applied. The probability

of random temporary emigration, that is, of being temporarily unavailable for detection, is confounded with the probability of detection given availability (Kendall, 1999). In other words, the detection parameter refers to the product of the probability of being available for detection and that of being detected, given being present. According to conventional wisdom, the interpretation of the occupancy parameter simply changes from the *probability of permanent presence* to the *probability of use sometime during the study period* (MacKenzie, 2005).

However, there may be cases when temporary emigration (dispersal) is so strong as to make the resulting estimates of probability of use meaningless, for example, effectively 1. In other cases, temporary emigration may be Markovian: whether a site is occupied at time  $t = 2$  depends on whether it was so at  $t = 1$ . Probability of (un-)availability is then no longer confounded with the probability of detection given availability, and naive application of single-season occupancy models results in biased estimates of occupancy (Kendall, 1999; Rota et al., 2009).

As a remedy, the  $J$  survey occasions may be assigned to subgroups and closure assumed only within each such subgroup. Owing to the seasonality of nature in most parts of the world, seasons over a series of years represent an extremely common, natural grouping factor. As an example, for birds or amphibians, replicate surveys are often conducted during the breeding season and this may be repeated over multiple years. Such a sampling at two temporal scales is called the robust design (Williams et al., 2002); each year, or breeding season, is called a primary sampling occasion, and the surveys within each season are called secondary sampling occasions. It is natural then to assume closure among secondary seasons only, that is, within each primary season, and allow change in the occurrence state among primary seasons. In the context of site-occupancy models, we then have observations from  $R$  sites,  $J$  replicate surveys (secondary sampling occasions), and  $K$  primary seasons (such as years), yielding detection/nondetection data  $y_{i,j,k}$  for site  $i$ , within-season survey  $j$ , and season  $k$ . Note that up to now in this chapter, index  $j$  was for all occasions, while in this section  $j$  will index secondary occasions only.

Given our expectation that occupancy changes among seasons  $k$ , how should we model occupancy dynamics? It would be simplest to treat season as a group and fit separate parameters for each, as we did in Section 12.3 in the context of abundance estimation in an open population. This is a reasonable approach, but there may be two issues with it. First, it treats observations from a site surveyed in different seasons as independent. However, whether a site is occupied at one time may depend on whether it was occupied previously, violating the independence assumption and representing a form of pseudoreplication (Hurlbert, 1984). This may result in too short standard error estimates, so it may be desirable to account for the repeated-measures nature of

multiseason data. Second, the interest of a study may focus on the parameters that govern occupancy dynamics, that is, colonization and extinction/survival. Occupancy is the quantity that metapopulation ecologists also call incidence (Hanski, 1994). Rather than simply describing changes of incidence over time, a metapopulation ecologist is interested in estimating probabilities of patch survival (or extinction) and patch colonization. This provides us with the motivation to explicitly model occupancy dynamics in terms of parameters describing the demographic components of that dynamics. This is achieved by the multiseason, or dynamic, site-occupancy model of MacKenzie et al. (2003). Moving from a single-season to a dynamic site-occupancy model is analogous to moving from a closed capture–recapture model (Chapter 6) to a Jolly–Seber model (Chapter 10) or from a classic binomial mixture model (Chapter 12) to the generalized binomial mixture model of Dail and Madsen (2011).

To describe detection/nondetection data  $y_{i,j,k}$  for site  $i$  and (within-season) replicate survey  $j$  in season  $k$ , we follow the hierarchical, or state-space, formulation of the model by Royle and Kéry (2007). We describe the observed data in a two-level random-effects model, that is, as a set of two linked stochastic processes or equations. The first equation describes the ecological process, that is, the evolution of the latent occurrence state  $z_{i,k}$  of site  $i$  over season  $k$ . Occurrence is latent because it is only partly observable and hence must be estimated from the observations  $y_{i,j,k}$ . The second equation describes the observation process, that is, the mapping of the latent state  $z_{i,k}$  on observation  $y_{i,j,k}$ . The basic model is thus the following:

$$\begin{aligned} z_{i,k} &\sim \text{Bernoulli}(\psi_{i,k}) & 1. \text{ Ecological process yields true state} \\ y_{i,j,k} | z_{i,k} &\sim \text{Bernoulli}(z_{i,k} p_{i,j,k}) & 2. \text{ Observation process yields observations} \end{aligned}$$

The sole change to the single-season occupancy model is the addition of an index for season,  $k$ . The model now describes the latent occurrence state  $z_{i,k}$  at site  $i$  in season  $k$  as a Bernoulli trial with occupancy parameter  $\psi_{i,k}$ . Observation  $y_{i,j,k}$  is equal to 1 if a species is detected during temporal replicate  $j$  at site  $i$  in season  $k$ , and zero otherwise, and is another Bernoulli trial governed by the product of the occurrence state at  $i$  and  $k$  and detection probability  $p_{i,j,k}$ .

As said above, we could model  $y_{i,j,k}$  by simply treating season  $k$  as a group, which would be equivalent to fitting separate occupancy models to the data from each season. This is how we modeled changes in abundance over multiple seasons in Section 12.3. But now, we will describe the state dynamics in an explicit, Markovian way instead: we will specify an initial state and two sets of parameters that govern subsequent changes in a first-order autoregressive manner. This is a simple extension of the ecological process model above. For clarity, we will drop the site index ( $i$ ).

$$\begin{array}{ll} z_1 \sim \text{Bernoulli}(\psi_1) & \text{1a. Initial ecological state in first season} \\ z_{k+1} | z_k \sim \text{Bernoulli}(z_k \phi_k + (1 - z_k) \gamma_k) & \text{1b. Markovian transitions in later seasons} \end{array}$$

Hence, in season 1, occurrence is a simple Bernoulli trial as before. In all later seasons, the occurrence state  $z_{k+1}$  of a site in season  $k + 1$  is a Bernoulli trial with a success parameter that depends on two things: whether the site was occupied at time  $k$  and on the value of either a survival or a colonization parameter. Hence, if a site was occupied during season  $k$  (i.e.,  $z_k = 1$  and therefore,  $1 - z_k = 0$ ), it will be re-occupied in the following season with probability  $\phi_k$ ; this is the (site) survival probability. Of course, we could equivalently describe this in terms of the complement of survival, extinction probability  $1 - \phi_k$ . On the other hand, if a site was unoccupied during season  $k$  (i.e.,  $z_k = 0$  and therefore,  $1 - z_k = 1$ ), it will be occupied at  $k + 1$  with probability  $\gamma_k$ ; this is the (site) colonization probability.

The state process of the dynamic site-occupancy model is exactly equivalent to a classical metapopulation model (Hanski, 1998), which expresses changes between time  $t$  and  $t + 1$  in the occurrence state of a collection of patches as a function of the probabilities of colonization of patches unoccupied at time  $k$ , and of survival (or alternatively, of extinction) of patches that were occupied at time  $k$ . This model makes the important assumption that the occurrence state of each patch can be determined perfectly, that is, that detection probability is equal to 1. Dynamic site-occupancy models represent an extended metapopulation model: the extension lies in an explicit accounting for imperfect detection (MacKenzie et al., 2003; Royle and Kéry, 2007), which becomes possible whenever replicated detection/nondetection observations are available within single periods of closure for at least some sites and/or such periods. Not accounting for imperfect detection in conventional metapopulation models will lead to biased estimates of *all* estimated quantities: incidence will be estimated too low and the probabilities of extinction, colonization, and turnover will all be estimated too high (Moilanen (2002); Royle and Dorazio (2008); see also Risk et al. (2011), for a robust-design incidence function model).

We will next simulate a data set under the dynamic site-occupancy model and analyze that. Afterwards, we will analyze a real data set. You will find another example of a dynamic occupancy model in the OpenBUGS manual (Examples > Ecology examples > Sparrowhawks).

### 13.5.1 Generation and Analysis of Simulated Data

We assume that we have data from a typical population study of a (nocturnal) bird of prey, the Long-eared owl (Fig. 13.9). Each of a total of  $R$  territories was surveyed on  $J$  occasions during each of  $K$  breeding seasons (years), and it was recorded whether any sign of territory occupation was



**FIGURE 13.9** Long-eared owl (*Asio otus*), Finland, 2008 (Photograph by T. Muukkonen).

detected. Our data  $y_{i,j,k}$  represent detection ( $y = 1$ ) or nondetection ( $y = 0$ ) of an owl in territory  $i$ , during replicate survey  $j$  in breeding season (year)  $k$ . Note that here, occupancy is equivalent to abundance because the number of occupied sites is exactly the local population size of owls.

We define a function to generate a data set. As always, apart from generating a data set to be analyzed later, this function may be used to get insights into the structure of the model used to analyze the data, issues of parameter estimation, or required samples sizes (see Section 1.5).

```

data.fn <- function(R = 250, J = 3, K = 10, psil = 0.4, range.p =
  c(0.2, 0.4), range.phi = c(0.6, 0.8), range.gamma = c(0, 0.1)) {
  # Function to simulate detection/nondetection data for dynamic
  # site-occ model
  # Annual variation in probabilities of patch survival, colonization and
  # detection is specified by the bounds of a uniform distribution.
  # Function arguments:
  # R - Number of sites
  # J - Number of replicate surveys
  # K - Number of years
  # psil - occupancy probability in first year
  # range.p - bounds of uniform distribution from which annual p drawn
  # range.psi and range.gamma - same for survival and colonization
  # probability
}

```

```

# Set up some required arrays
site <- 1:R                      # Sites
year <- 1:K                        # Years
psi <- rep(NA, K)                  # Occupancy probability
muZ <- z <- array(dim=c(R, K))    # Expected and realized occurrence
y <- array(NA, dim=c(R, J, K))     # Detection histories

# Determine initial occupancy and demographic parameters
psi[1] <- psi1                      # Initial occupancy probability
p <- runif(n=K, min=range.p[1], max=range.p[2])
phi <- runif(n=K-1, min=range.phi[1], max=range.phi[2])
gamma <- runif(n=K-1, min=range.gamma[1], max=range.gamma[2])

# Generate latent states of occurrence
# First year
z[,1] <- rbinom(R, 1, psi[1])      # Initial occupancy state

# Later years
for(i in 1:R){                      # Loop over sites
  for(k in 2:K){                    # Loop over years
    muZ[k] <- z[i, k-1]*phi[k-1] + (1-z[i, k-1])*gamma[k-1]
    # Prob for occ.
    z[i,k] <- rbinom(1, 1, muZ[k])
  } #k
} #i

# Plot realised occupancy
plot(year, apply(z, 2, mean), type = "l", xlab = "Year", ylab =
  "Occupancy or Detection prob.", col = "red", xlim = c(0, K+1),
  ylim = c(0,1), lwd = 2, lty = 1, frame.plot = FALSE, las = 1)
lines(year, p, type = "l", col = "red", lwd = 2, lty = 2)

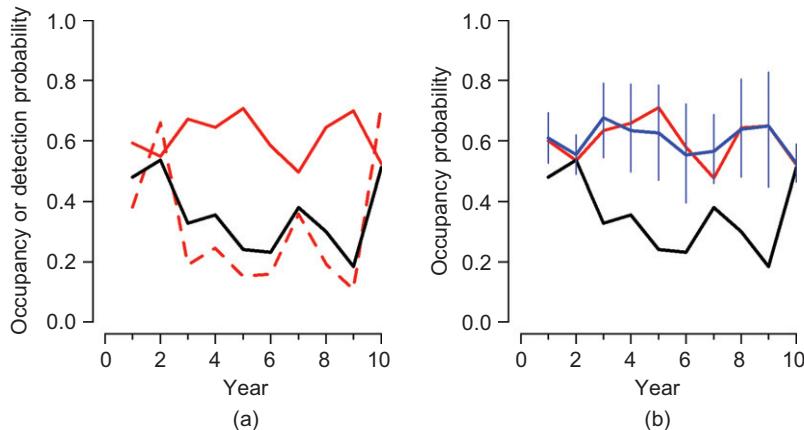
# Generate detection/nondetection data
for(i in 1:R){
  for(k in 1:K){
    prob <- z[i,k] * p[k]
    for(j in 1:J){
      y[i,j,k] <- rbinom(1, 1, prob)
    } #j
  } #k
} #i

# Compute annual population occupancy
for (k in 2:K){
  psi[k] <- psi[k-1]*phi[k-1] + (1-psi[k-1])*gamma[k-1]
}

# Plot apparent occupancy
psi.app <- apply(apply(y, c(1,3), max), 2, mean)
lines(year, psi.app, type = "l", col = "black", lwd = 2)
text(0.85*K, 0.06, labels = "red solid - true occupancy\n red
dashed - detection\n black - observed occupancy")

# Return data
return(list(R=R, J=J, K=K, psi=psi, psi.app=psi.app, z=z,
           phi=phi, gamma=gamma, p=p, y=y))
}

```



**FIGURE 13.10** (a) Simulated territory occupancy data for long-eared owls. Truth is shown in red (solid, occupancy probability; dashed, detection probability) and the observed occupancy probability in black. The difference between the red and the black lines is due to detection error. (b) Comparison between true, observed, and estimated occupancy probability. Truth is shown in red, estimates under the site-occupancy model (with 95% CRI) are in blue, and naïve estimates (observed values) are in black. (Note: Using the R code in the book, you will generate each plot separately.)

We execute the function once to obtain a data set for 250 owl territories with three surveys in each of 10 years (Fig. 13.10a).

```
data <- data.fn(R = 250, J = 3, K = 10, psi1 = 0.6, range.p = c(0.1, 0.9),
range.phi = c(0.7, 0.9), range.gamma = c(0.1, 0.5))
```

We attach the data set and produce a simple summary.

```
attach(data)
str(data)
> str(data)
List of 10
 $ R      : num 250
 $ J      : num 3
 $ K      : num 10
 $ psi    : num [1:10] 0.6 0.535 0.635 0.658 0.71 ...
 $ psi.app: num [1:10] 0.48 0.536 0.328 0.356 0.24 0.232 0.38 0.3 0.184 0.512
 $ z      : num [1:250, 1:10] 0 1 0 1 1 1 1 0 1 ...
 $ phi   : num [1:9] 0.791 0.761 0.805 0.879 0.772 ...
 $ gamma : num [1:9] 0.151 0.489 0.403 0.384 0.105 ...
 $ p     : num [1:10] 0.382 0.659 0.19 0.246 0.151 ...
 $ y     : num [1:250, 1:3, 1:10] 0 0 0 1 1 0 1 0 0 0 ...
```

We conduct the analysis using code from Royle and Kéry (2007), which includes the estimation of the actual number of occupied territories (among the 250), the occupancy-based population growth rate, and the turnover rate.

```

# Specify model in BUGS language
sink("Dynocc.txt")
cat("
model {

# Specify priors
psi1~dunif(0, 1)
for (k in 1:(nyear-1)){
  phi[k] ~ dunif(0, 1)
  gamma[k] ~ dunif(0, 1)
  p[k] ~ dunif(0, 1)
}
p[nyear] ~ dunif(0, 1)

# Ecological submodel: Define state conditional on parameters
for (i in 1:nsite){
  z[i,1] ~ dbern(psi1)
  for (k in 2:nyear){
    muZ[i,k]<- z[i,k-1]*phi[k-1] + (1-z[i,k-1])*gamma[k-1]
    z[i,k] ~ dbern(muZ[i,k])
  } #k
} #i

# Observation model
for (i in 1:nsite){
  for (j in 1:nrep){
    for (k in 1:nyear){
      muy[i,j,k] <- z[i,k]*p[k]
      y[i,j,k] ~ dbern(muy[i,j,k])
    } #k
  } #j
} #i

# Derived parameters: Sample and population occupancy, growth rate
# and turnover
psi[1] <- psi1
n.occ[1]<-sum(z[1:nsite,1])
for (k in 2:nyear){
  psi[k] <- psi[k-1]*phi[k-1] + (1-psi[k-1])*gamma[k-1]
  n.occ[k] <- sum(z[1:nsite,k])
  growthr[k] <- psi[k]/psi[k-1]
  turnover[k-1] <- (1 - psi[k-1]) * gamma[k-1]/psi[k]
}
}

",fill=TRUE)
sink()

# Bundle data
win.data<- list(y=y, nsite=dim(y)[1], nrep=dim(y)[2], nyear=dim(y)[3])

# Initial values
Zst <- apply(y, c(1, 3), max) # Observed occurrence as inits for z
inits <- function() { list(z=zst) }

# Parameters monitored
params <- c("psi", "phi", "gamma", "p", "n.occ", "growthr",
"turnover")

```

```

# MCMC settings
ni <- 2500
nt <- 4
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT 3 min)
out <- bugs(win.data, inits, params, "Dynocc.txt", n.chains = nc,
            n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
            bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out, dig = 2)
[...]
          mean     sd   2.5%   25%   50%   75% 97.5% Rhat n.eff
psi[1]    0.61  0.04  0.53  0.58  0.61  0.64  0.69  1.00  720
[...]
psi[10]   0.53  0.03  0.46  0.50  0.53  0.55  0.59  1.00  810
phi[1]    0.83  0.04  0.75  0.81  0.83  0.86  0.90  1.00  1500
[...]
phi[9]    0.67  0.06  0.55  0.63  0.68  0.71  0.80  1.00  1500
gamma[1]   0.12  0.06  0.01  0.08  0.12  0.17  0.25  1.00  1500
[...]
gamma[9]   0.24  0.12  0.02  0.15  0.24  0.33  0.44  1.00  1500
p[1]      0.40  0.03  0.35  0.38  0.40  0.42  0.47  1.00  1100
[...]
p[10]    0.71  0.03  0.66  0.69  0.71  0.72  0.75  1.00  1500
n.occ[1]  152.32 7.48 138.00 147.00 152.00 158.00 166.00 1.00  1500
[...]
n.occ[10] 131.51 2.18 128.00 130.00 131.00 133.00 137.00 1.00  1100
growthr[2] 0.91  0.06  0.80  0.87  0.91  0.95  1.04  1.00  1500
[...]
growthr[10] 0.83  0.14  0.62  0.73  0.81  0.90  1.16  1.00  1400
turnover[1] 0.09  0.05  0.01  0.05  0.08  0.12  0.20  1.00  1500
[...]
turnover[9] 0.17  0.11  0.01  0.09  0.16  0.24  0.41  1.00  1500
[...]

```

We compare truth and estimates of truth (posterior mean, sd, and 95% CRI) in tables ...

```

print(cbind(data$psi, out$summary[1:K, c(1, 2, 3, 7)]), dig = 3)
print(cbind(data$phi, out$summary[(K+1):(K+(K-1)), c(1, 2, 3, 7)]),
      dig = 3)
print(cbind(data$gamma, out$summary[(2*K):(2*K+(K-2)), c(1, 2, 3,
    7)]), dig = 3)
print(cbind(data$p, out$summary[(3*K-1):(4*K-2), c(1, 2, 3, 7)]),
      dig = 3)

```

... and in a picture (Fig. 13.10b).

```

plot(1:K, data$psi, type = "l", xlab = "Year", ylab = "Occupancy
probability", col = "red", xlim = c(0, K+1), ylim = c(0, 1), lwd = 2,
lty = 1, frame.plot = FALSE, las = 1)

```

```
lines(1:K, data$psi.app, type = "l", col = "black", lwd = 2)
points(1:K, out$mean$psi, type = "l", col = "blue", lwd = 2)
segments(1:K, out$summary[1:K, 3], 1:K, out$summary[1:K, 7],
          col = "blue", lwd = 1)
```

We are rather satisfied with the performance of the metapopulation estimators of the model.

### 13.5.2 Dynamic Occupancy Modeling in a Real Data Set

As another illustration of the dynamic occupancy model of MacKenzie et al. (2003), we will use data from the Six-spot burnet (Fig. 13.11) collected in the Swiss butterfly monitoring program. Remember that we have 95 sites with two replications in each of 7 seasons, and that a “season”



**FIGURE 13.11** The Six-spot burnet *Zygaena filipendulae*, a day-flying moth, Switzerland, 2004 (Photograph by T. Marent).

represents one day, within which a transect is surveyed back and forth (for further description, see Section 12.3; Kéry et al. (2009b); Dorazio et al. (2010)). This is a more typical example of an occupancy model, where a “site” represents a 2.5 km transect in a 1 km<sup>2</sup> square and is so large relative to the space requirements of the study species that it can be inhabited by many (hundred) individuals. Thus, there is no longer a 1:1 relationship between occupancy and abundance as in the owl example.

After reading the count data into R, we will first reformat the data into a 3-dimensional array, as we did for the multiseason binomial mixture model in Section 12.3. We start with a format where butterfly counts from different “seasons” (days) are stacked. For this code to work, the data must be balanced, that is, we must have the same number of surveyed sites in each “season” (day). This is not a requirement of the model, simply of our code. If you have variation in the number of sites surveyed, then you have to “fill in” the data using NAs to make them balanced or else vectorize the BUGS model description (see chapter 21 in Kéry, 2010).

```
# Read in the data and put it into 3D array
bdat <- read.table(file = "burnet.txt", header = T)
str(bdat)

y <- array(NA, dim=c(95, 2, 7)) # 95 sites, 2 reps, 7 days

for (i in 1:7) {
  sel.rows <- bdat$day == i
  y[,,i] <- as.matrix(bdat)[sel.rows, 3:4]
}
str(y)

# Convert counts to detection/non-detection data
y[y>0] <- 1

# Look at the number of sites with detections for each day
tmp <- apply(y, c(1,3), max, na.rm = TRUE)
tmp[tmp == "-Inf"] <- NA
apply(tmp, 2, sum, na.rm = TRUE)
[1] 0 0 3 10 17 17 6
```

There are no detections of burnets at all during the first two days. We are now ready to fit the dynamic occupancy model in WinBUGS. The code is the same as before (Section 13.5.1) so we simply recycle the BUGS model description from there.

```
# Bundle data
win.data <- list(y = y, nsite = dim(y)[1], nrep = dim(y)[2], nyear = dim(y)[3])

# Initial values
inits <- function() { list(z = apply(y, c(1, 3), max)) }

# Parameters monitored
params <- c("psi", "phi", "gamma", "p", "n.occ", "growthr",
  "turnover")
```

```

# MCMC settings
ni <- 5000
nt <- 4
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 1 min)
out1 <- bugs(win.data, inits, params, "Dynocc.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out1, dig = 3)

      mean     sd   2.5%   25%   50%   75%   97.5%   Rhat n.eff
psi[1]    0.106  0.208  0.001  0.009  0.025  0.078  0.895  1.048   60
[...]
psi[7]    0.117  0.052  0.044  0.080  0.106  0.142  0.246  1.005   490
phi[1]    0.442  0.294  0.014  0.177  0.419  0.687  0.969  1.007   870
[...]
phi[6]    0.447  0.183  0.170  0.313  0.415  0.551  0.900  1.002   1000
gamma[1]  0.151  0.230  0.001  0.014  0.050  0.171  0.905  1.025   100
[...]
gamma[6]  0.026  0.026  0.001  0.008  0.018  0.036  0.095  1.004   650
p[1]      0.294  0.296  0.002  0.038  0.179  0.497  0.948  1.037   79
[...]
p[7]      0.536  0.183  0.195  0.398  0.540  0.679  0.864  1.005   430
n.occ[1]  9.268  19.972 0.000  0.000  1.000  6.000  85.000 1.171   32
[...]
n.occ[7]  9.515  3.971  6.000  7.000  8.000 11.000  20.000 1.009   300
growthr[2] 21.130 257.481 0.078  0.841  2.037  6.563 117.412 1.026   88
[...]
growthr[7] 0.548  0.230  0.222  0.384  0.504  0.671  1.101  1.003   710
turnover[1] 0.714  0.287  0.054  0.539  0.826  0.953  0.998  1.011   260
[...]
turnover[6] 0.172  0.139  0.005  0.061  0.137  0.251  0.505  1.003   940

```

We see that some of the parameters associated with the first two days, when no burnets were observed, are not estimable. An indication of this is that their posterior distributions cover (almost) the entire range of their prior distributions, that is, the 95% CRI essentially covers the range from 0 to 1 for the probability parameters. This means that the data contain no information about these parameters. The parameters describing the dynamics of occupancy, survival ( $\phi$ ), colonization ( $\gamma$ ), and the growth rate, may all offer interesting insights into the factors that drive the population dynamics of a species in the context of occurrence.

Apart from the third day, when very few burnets were observed (and during the first two, see above), detection probability appears to be similar. Hence, we pool the detection parameters and fit a model with constant detection probability. In addition, as an exercise we aggregate the binary response over the two replicates per day and specify a binomial(2,  $p$ ) data distribution instead of a Bernoulli( $p$ ). When there

is no modeled structure among replicate surveys, this model parameterization is computationally more efficient than the one with a Bernoulli response.

```
# Specify model in BUGS language
sink("Dynocc2.txt")
cat("
model {

# Specify priors
psi1 ~ dunif(0, 1)
for (k in 1:(nyear-1)){
  phi[k] ~ dunif(0, 1)
  gamma[k] ~ dunif(0, 1)
}
p ~ dunif(0, 1)

# Both models at once
for (i in 1:nsite){
  z[i,1] ~ dbern(psi1)      # State model 1: Initial state
  for (k in 2:nyear){        # State model 2: State dynamics
    muZ[i,k] <- z[i,k-1]*phi[k-1] + (1-z[i,k-1])*gamma[k-1]
    z[i,k] ~ dbern(muZ[i,k])

    # Observation model
    muy[i,k] <- z[i,k]*p
    y[i,k] ~ dbin(muy[i,k], 2)
  } #k
} #i

# Derived parameters: Sample and population occupancy, growth
# rate and turnover
psi[1] <- psi1
n.occ[1] <- sum(z[1:nsite,1])
for (k in 2:nyear){
  psi[k] <- psi[k-1]*phi[k-1] + (1-psi[k-1])*gamma[k-1]
  n.occ[k] <- sum(z[1:nsite,k])
  growthr[k] <- psi[k]/psi[k-1]
  turnover[k-1] <- (1 - psi[k-1]) * gamma[k-1]/psi[k]
}
}

",fill=TRUE)
sink()

# Aggregate detections over reps within a day and bundle data
yy <- apply(y, c(1, 3), sum, na.rm=TRUE)
win.data <- list(y=yy, nsite=dim(yy)[1], nyear=dim(yy)[2])

# Initial values
inits <- function(){list(z=apply(y, c(1, 3), max))}

# Parameters monitored
params <- c("psi", "phi", "gamma", "p", "n.occ", "growthr",
"turnover")
```

```

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT 1 min)
out2 <- bugs(win.data, inits, params, "Dynocc2.txt", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE, bugs.directory =
  bugs.dir, working.directory = getwd())

# Summarize posteriors
print(out2, dig = 3)

      mean     sd   2.5%   25%   50%   75%   97.5%   Rhat n.eff
psi[1]    0.461  0.394  0.003  0.058  0.387  0.902  0.997  1.122    21
[...]
psi[7]    0.093  0.032  0.040  0.070  0.090  0.114  0.165  1.002  1700
phi[1]    0.121  0.207  0.001  0.008  0.026  0.121  0.778  1.087    28
[...]
phi[6]    0.368  0.120  0.154  0.280  0.362  0.447  0.625  1.001  3000
gamma[1]   0.108  0.202  0.001  0.007  0.020  0.083  0.790  1.070    33
[...]
gamma[6]   0.017  0.017  0.000  0.005  0.012  0.024  0.063  1.001  3000
p         0.646  0.059  0.525  0.608  0.648  0.687  0.756  1.001  3000
n.occ[1]   43.739 38.086 0.000 5.000 36.000 87.000 95.000 1.176    16
[...]
n.occ[7]   7.241  1.301  6.000  6.000  7.000  8.000 10.000 1.001  3000
growthr[2] 0.742  4.958  0.004  0.021  0.060  0.327  5.190  1.096    26
[...]
growthr[7] 0.433  0.139  0.196  0.334  0.422  0.517  0.742  1.002  1900
turnover[1] 0.507  0.300  0.016  0.244  0.507  0.777  0.981  1.002  1400
[...]
turnover[6] 0.143  0.126  0.004  0.048  0.108  0.206  0.464  1.001  3000

```

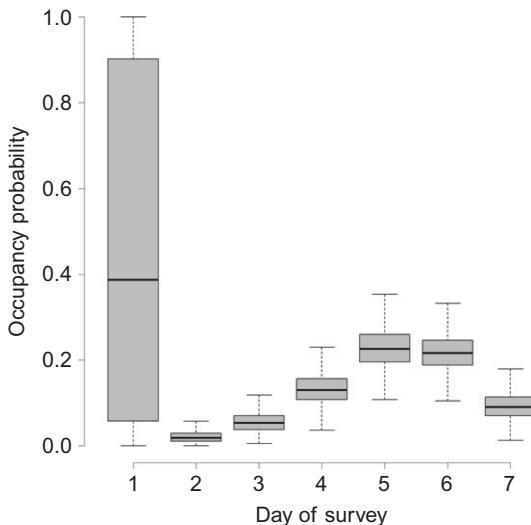
We plot what we have learnt about the occupancy, or incidence, of Swiss burnets over the season (Fig. 13.12).

```

DAY <- cbind(rep(1, out2$n.sims), rep(2, out2$n.sims), rep(3,
  out2$n.sims), rep(4, out2$n.sims), rep(5, out2$n.sims), rep(6,
  out2$n.sims), rep(7, out2$n.sims))
boxplot(out2$sims.list$psi ~ DAY, col = "gray", ylab = "Occupancy
probability", xlab = "Day of survey", las = 1, frame.plot = FALSE)

```

We see the typical unimodal phenology of an insect in temperate latitudes (Kéry et al., 2009). Six-spot burnets are most widespread in Switzerland during the period in which survey number 5 is made. Interestingly, although no burnets were seen during either the first or the second day, the posterior distribution for occupancy was quite different for the two days. There are two reasons for this. First, the Markovian model propagates information backwards in time and so occurrence at  $k=2$  ( $z_{i,2}$ ) is informed directly by  $z_{i,3}$  because there are data at  $k=3$ . Conversely,  $z_{i,1}$  gets no direct information



**FIGURE 13.12** Occupancy probability of the burnet over a season: summary of posterior distributions for survey day 1 through survey day 7. No burnets at all were seen during the first two days.

at all from  $k = 2$  because there were no observations (J.A. Royle, pers. comm.). Second, different amounts of information are available for occupancy on the two days. There were only 75 sites with surveys on the first, but 87 sites with surveys on the second day. As a consequence, the occupancy parameter was not estimable on the first day; the posterior samples simply reflected the prior. In contrast, on the second day, occupancy was estimated at effectively zero. You can compare the sample sizes for each day like the following:

```
apply(apply(y, c(1, 3), max), 2, function(x) {sum(!is.na(x))})
[1] 75 87 95 95 95 87
```

In summary, the dynamic site-occupancy model is a powerful extension to the classical metapopulation model. Depending on the definition of a site and the state of occurrence, dynamic occupancy models can be used to describe the dynamics of a vast array of systems. Covariates can be introduced for all parameters via the usual GLM link functions. The main challenge when applying the model may be a data management and parameter bookkeeping one: to put the data in the required multidimensional arrays and not to get confused with multidimensional model code.

## 13.6 MULTISTATE OCCUPANCY MODELS

So far, we have been treating occurrence as a binary variable. However, frequently we can distinguish different states of occurrence. Examples include “single bird”, “nonreproductive pair”, and “reproductive pair”

when studying territory occupancy, “breeding possible”, “breeding probable”, or “breeding confirmed” in bird atlas studies (Schmid et al., 1998), different population size classes in the monitoring of vocal amphibians (Royle and Link, 2005), or “occupied by species A only”, “occupied by species B only”, or “occupied by both species” in studies of species interactions. Apart from detection uncertainty, there is an additional potential component of uncertainty in these examples: state uncertainty, that is, whether a site observed in one state truly is in that state. For example, when species A is observed at a site, the true state of that site could either be “occupied by species A only” or “occupied by both species”.

The multistate site-occupancy model is used for inference about multiple states of occupancy in the presence of both state and detection uncertainty. This model seems to have been independently developed by Royle and Link (2005) and Nichols et al. (2007), providing another example for the independent and (more or less) simultaneous development of a model, such as the Cormack-Jolly-Seber model (Cormack, 1964; Jolly, 1965; Seber, 1965), single-state site-occupancy models (MacKenzie et al., 2002; Tyre et al., 2003), and spatial capture–recapture models (Borchers and Efford, 2008; Royle and Young, 2008). The explicit merging of site-occupancy models with multistate models (Chapter 9) holds promise because the combination of two already very general model classes likely results in even more flexible models. It is likely that many ideas that are well understood and applied in the multistate arena may be taken over to site-occupancy models as well.

In the following, we focus on the simplest possible multistate occupancy model, where two occurrence states along with the third state “unoccupied” are distinguished in a closed population. The generalization to more than two states is straightforward. We illustrate with data from a survey of long-eared owls (Fig. 13.9), where either hooting adult males or begging young are detected, or nothing at all. If a hooting male is heard, we are unsure about whether reproduction is taking place at a site. If we fail to hear anything, we are unsure about whether a site is occupied at all as well as whether there is reproduction. In contrast, when hearing begging young, there is no uncertainty about state and occurrence.

The development of the model is nearly identical to that of the multistate model (Chapter 9). First, we need to define lists of true and of observed states. The true states in our example are “not occupied”, “occupied without reproduction”, and “occupied with reproduction”. The list of observed states comprises “not seen”, “seen without reproduction”, and “seen with reproduction”. A hierarchical model for data from this system distinguishes a description of the state and another of the observation process. We therefore introduce the latent variable  $z$ , which defines the true state of each site and can take values 1 (site not occupied), 2 (site occupied without reproduction), or 3 (site occupied

with reproduction). The probability of the state of site  $i$  is modeled using a categorical distribution as

$$z_i \sim \text{categorical}(\Omega_i),$$

where  $\Omega_i$  is the state vector. The state vector has as many elements as there are states and each element is the probability that site  $i$  is in a given state. In our example, we have

$$\Omega_i = \begin{bmatrix} 1 - \psi_{1,i} - \psi_{2,i} \\ \psi_{1,i} \\ \psi_{2,i} \end{bmatrix},$$

where  $\psi_{1,i}$  is the probability that site  $i$  is occupied without reproduction,  $\psi_{2,i}$  is the probability that site  $i$  is occupied and reproduction takes place, and  $1 - \psi_{1,i} - \psi_{2,i}$  is the probability that site  $i$  is unoccupied. Obviously, the three probabilities need to sum to 1, and we often assume that the probabilities are the same at all sites (no index  $i$ ).

Given the true state  $z_i$  of site  $i$ , the observation process links the true state with the observations  $(y_{i,j})$ . We write

$$y_{i,j} | z_i \sim \text{categorical}(\Theta_{z_i, 1 \dots O, j}),$$

where  $\Theta$  is the observation array and  $O$  is the number of observed states. The array has four dimensions; the last two refer to site ( $i$ ) and survey ( $j$ ). If detection is assumed to be the same at all sites and constant among surveys, the observation array becomes a two-dimensional matrix. The first dimension refers to the true state and the second to the observed states. The elements of the matrix are the probabilities of an observation given a state. Assuming constancy over sites and surveys, the most general observation matrix  $\Theta$  is

	not seen	seen without rep.	seen with rep.
not occupied	$\pi_{1,1}$	$\pi_{1,2}$	$\pi_{1,3}$
occupied without reproduction	$\pi_{2,1}$	$\pi_{2,2}$	$\pi_{2,3}$
occupied with reproduction	$\pi_{3,1}$	$\pi_{3,2}$	$\pi_{3,3}$

The true states are in the rows and the observed states in the columns. Thus,  $\pi_{m,k}$  denotes the probability of classifying a site in state  $m$  as being in state  $k$ . These probabilities are either detection or genuine classification probabilities or both. Clearly, the probabilities of correct classification are in the diagonal, while the off-diagonals contain the probabilities of incorrect classification. The matrix is row-stochastic, so the three probabilities in the same row are not independent; rather, they sum to one.

This matrix defines the most general multistate model that could be fitted in a site-occupancy context. Given sufficient data, all parameters should be estimable. However, frequently, there is a natural order in the modeled states and some errors are unlikely or impossible. Typically, it can be assumed that classification errors only occur in one direction, so that a “higher” state can be erroneously taken to be a lower state, but not the other way round. For instance, a site with reproduction could be classified as having no reproduction if only an adult is heard hooting and no begging young are heard, but not the other way round. The result of this is that we model a restricted version of the fully general observation matrix (now we also make explicit the relationships among cell probabilities within a row):

	not seen	seen without rep.	seen with rep.
not occupied	1	0	0
occupied without reproduction	$1 - \pi_{2,2}$	$\pi_{2,2}$	0
occupied with reproduction	$1 - \pi_{3,2} - \pi_{3,3}$	$\pi_{3,2}$	$\pi_{3,3}$

Both Royle and Link (2005) and Nichols et al. (2007) describe restricted models of this kind, where a site in state 1 (unoccupied) can only be observed in state 1 (we assume there are no false positives), but sites in state 2 can be observed in state 1 or 2 and sites in state 3 in all three states (1, 2, or 3).

This model can be re-expressed in various parameterizations. What this means is that the elements of the state vector and the elements  $\pi_{m,k}$  in the observation matrix can be rewritten as functions of other parameters that may be more interesting biologically or that may allow a more natural formulation of covariate effects; see Royle and Link (2005) and Nichols et al. (2007). Our parameterization in this chapter is as follows:

State vector	Observation matrix
$\begin{bmatrix} 1-\psi \\ \psi(1-r) \\ \psi r \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1-p_2 & p_2 & 0 \\ p_{3,1} & p_{3,2} & p_{3,3} \end{bmatrix}$

Here,  $\psi$  is the probability of occupancy, regardless of reproduction, and  $r$  is the probability that reproduction takes place at an occupied site. In the observation matrix,  $p_2$  is the detection probability of a site without reproduction,  $p_{3,3}$  is the probability that at a site with reproduction, the species is detected and reproduction is observed (i.e., the state is correctly classified),  $p_{3,2}$  is the probability that at a site with reproduction, the species is

detected but reproduction is not observed (i.e., the state is misclassified), and  $p_{3,1}$  is the probability that the species is not detected at a site with reproduction. The three probabilities,  $p_{3,k}$  must sum to one, and this is accounted for by our choice of a Dirichlet prior in the BUGS model description; see below.

To illustrate this model, we use data on territory occupancy of the long-eared owl (Fig. 13.9) from a long-term population study of our colleague Simon Birrer at the Swiss Ornithological Institute. Birrer has been surveying 40 owl territories repeatedly in every breeding season since 1989. Not all sites were checked in every year and we chose the data from 2009, when 31 sites were checked up to 5 times. We read in the data and briefly look at them.

```
owls <- read.table("owls.txt", header = TRUE)
str(owls)
```

The variables entitled `obs1`-`obs5` denote the result of each survey: detection of no owl at all (0), of a hooting owl (1) or of begging young (2). The variables entitled `date1`-`date5` give the Julian date of each survey. To fit the model, we must relabel the states because WinBUGS does not allow indices of 0. Hence, we denote the states in the same way as defined above. This relabeling is done in the data bundle statement below.

We specify the model with default vague priors for all parameters. The beta terms are used to specify a vague Dirichlet prior for the multinomial distribution represented by row three in the observation matrix above (see also Section 9.6). Our model could accommodate time variation in the observation matrix, but at first we will assume constancy of parameters over time.

```
# Specify model in BUGS language
sink("modell1.txt")
cat("
model {
  # Priors
  p2 ~ dunif(0, 1)
  psi ~ dunif(0, 1)
  r ~ dunif(0, 1)
  for (i in 1:3) {
    beta[i] ~ dgamma(1, 1) # Induce Dirichlet prior
    p3[i] <- beta[i]/sum(beta[])
  }
  # Define state vector
  for (s in 1:R) {
    phi[s,1] <- 1 - psi           # Prob. of nonoccupation
    phi[s,2] <- psi * (1 - r)      # Prob. of occupancy without repro
    phi[s,3] <- psi * r            # Prob. of occupancy and repro
  }
}
```

```

# Define observation matrix
# Order of indices: true state, time, observed state
for (t in 1:T) {
  p[1,t,1] <- 1
  p[1,t,2] <- 0
  p[1,t,3] <- 0
  p[2,t,1] <- 1-p2
  p[2,t,2] <- p2
  p[2,t,3] <- 0
  p[3,t,1] <- p3[1]
  p[3,t,2] <- p3[2]
  p[3,t,3] <- p3[3]
}

# State-space likelihood
# State equation: model of true states (z)
for (s in 1:R) {
  z[s] ~ dcat(phi[s,.])
}

# Observation equation
for (s in 1:R) {
  for (t in 1:T) {
    y[s,t] ~ dcat(p[z[s],t,.])
    } #t
  } #s

# Derived quantities
for (s in 1:R) {
  occ1[s] <- equals(z[s], 1)
  occ2[s] <- equals(z[s], 2)
  occ3[s] <- equals(z[s], 3)
}
n.occ[1] <- sum(occ1[]) # Sites in state 1
n.occ[2] <- sum(occ2[]) # Sites in state 2
n.occ[3] <- sum(occ3[]) # Sites in state 3
}
", fill=TRUE)
sink()

```

We analyze rows 2–6 in the `owls` data frame and convert them to a matrix called `y`.

```

# Bundle data
y <- as.matrix(owls[, 2:6])
y <- y + 1
win.data <- list(y=y, R=dim(Y)[1], T=dim(Y)[2])

# Initial values
zst <- apply(y, 1, max, na.rm=TRUE)
zst[zst == "-Inf"] <- 1
inits <- function(){list(z=zst)}

# Parameters monitored
params <- c("p2", "p3", "r", "psi", "n.occ") # Might want to add "z"

```

```

# MCMC settings
ni <- 2500
nt <- 2
nb <- 500
nc <- 3

# Call WinBUGS from R (BRT <1 min)
out1 <- bugs(win.data, inits, params, "modell.txt", n.chains = nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug =TRUE, bugs.directory =
  bugs.dir, working.directory =getwd())

# Summarize posteriors
print(out1, dig = 2)

      mean    sd   2.5%   25%   50%   75%  97.5%   Rhat n.eff
p2     0.35  0.19  0.04  0.21  0.33  0.46  0.82  1.00  2400
p3[1]  0.55  0.12  0.30  0.47  0.56  0.64  0.77  1.00  3000
p3[2]  0.21  0.09  0.05  0.14  0.20  0.26  0.40  1.00  530
p3[3]  0.24  0.12  0.07  0.16  0.22  0.31  0.51  1.00  550
r      0.64  0.21  0.24  0.48  0.64  0.81  0.98  1.01  350
psi    0.52  0.15  0.28  0.42  0.50  0.61  0.86  1.00  1900
n.occ[1] 19.11  5.39  5.00 16.00 20.00 23.00 27.00  1.00 1100
n.occ[2]  7.51  5.46  0.00  3.00  7.00 10.00 21.03  1.01  260
n.occ[3] 13.38  5.10  6.00 10.00 13.00 16.00 25.00  1.00  600

```

We estimate that 52% of sites are occupied, of which 64% by reproductive owls. For our specific sample of 40 sites, this translates into an estimated 13.4 occupied sites with and 7.5 sites without reproduction and 19.1 unoccupied sites. Detection probability of a site without reproduction is estimated at 0.35 and for a site with reproduction at 0.24. There is a probability of 0.21 to detect only hooting adults at a site with reproduction and one of 0.55 to miss it altogether. The parameters describing state uncertainty and detection error all refer to a single survey.

This model assumes that all parameters are constant, but the surveys take place over an extended time period (early March–early September), so this assumption may be unlikely. For instance, begging young will not be available over the entire period. Therefore, a more realistic model may be one that allows for these parameters to vary by occasion (i.e., survey 1–5).

```

# Specify model in BUGS language
sink("model2.txt")
cat("
model {

# Priors
psi~dunif(0, 1)
r~dunif(0,1 )

for (t in 1:T){
  p2[t] ~ dunif(0, 1)
  for (i in 1:3) {
    beta[i,t] ~ dgamma(1, 1)      # Induce Dirichlet prior
    p3[i,t] <- beta[i,t]/sum(beta[,t])
    } #i
  } #t
}
```

```

# Define state vector
for (s in 1:R) {
  phi[s,1] <- 1 - psi           # Prob. of nonoccupation
  phi[s,2] <- psi * (1 - r)     # Prob. of occupancy without repro.
  phi[s,3] <- psi * r          # Prob. of occupancy and repro.
}

# Define observation matrix
# Order of indices: true state, time, observed state
for (t in 1:T) {
  p[1,t,1] <- 1
  p[1,t,2] <- 0
  p[1,t,3] <- 0
  p[2,t,1] <- 1-p2[t]
  p[2,t,2] <- p2[t]
  p[2,t,3] <- 0
  p[3,t,1] <- p3[1,t]
  p[3,t,2] <- p3[2,t]
  p[3,t,3] <- p3[3,t]
}

# State-space likelihood
# State equation: model of true states (z)
for (s in 1:R) {
  z[s] ~ dcat(phi[s,])
}

# Observation equation
for (s in 1:R) {
  for (t in 1:T) {
    y[s,t] ~ dcat(p[z[s],t,])
    } #t
} #s

# Derived quantities
for (s in 1:R) {
  occ1[s] <- equals(z[s], 1)
  occ2[s] <- equals(z[s], 2)
  occ3[s] <- equals(z[s], 3)
}
n.occ[1] <- sum(occ1[]) # Sites in state 1
n.occ[2] <- sum(occ2[]) # Sites in state 2
n.occ[3] <- sum(occ3[]) # Sites in state 3
}
",fill=TRUE)
sink()

```

We recycle the remaining “ingredients” for the call to `bugs()` below.

```

# Call WinBUGS from R (BRT 1 min)
out2 <- bugs(win.data, inits, params, "model2.txt", n.chains=nc,
  n.thin=nt, n.iter=ni, n.burnin=nb, debug =TRUE, bugs.directory =
  bugs.dir, working.directory =getwd())

```

```
# Summarize posteriors
print(out2, dig = 2)
      mean    sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
p2[1]  0.76  0.19  0.32  0.65  0.80  0.92  0.99  1.01  440
p2[2]  0.57  0.21  0.17  0.41  0.57  0.72  0.94  1.00  3000
p2[3]  0.16  0.16  0.00  0.05  0.12  0.23  0.58  1.00  1600
p2[4]  0.34  0.20  0.04  0.19  0.32  0.47  0.78  1.01  670
p2[5]  0.27  0.21  0.01  0.09  0.22  0.39  0.78  1.00  3000
p3[1,1] 0.53  0.17  0.20  0.40  0.53  0.65  0.84  1.00  1300
p3[1,2] 0.33  0.17  0.06  0.20  0.32  0.44  0.68  1.00  3000
p3[1,3] 0.41  0.19  0.08  0.26  0.40  0.55  0.80  1.00  3000
p3[1,4] 0.53  0.22  0.11  0.36  0.54  0.70  0.91  1.00  3000
p3[1,5] 0.37  0.25  0.02  0.16  0.34  0.55  0.87  1.00  3000
p3[2,1] 0.37  0.16  0.09  0.25  0.36  0.48  0.70  1.00  3000
p3[2,2] 0.14  0.12  0.00  0.04  0.10  0.19  0.44  1.00  1300
p3[2,3] 0.15  0.13  0.00  0.04  0.11  0.21  0.48  1.00  1000
p3[2,4] 0.24  0.19  0.01  0.09  0.20  0.36  0.69  1.00  3000
p3[2,5] 0.31  0.23  0.01  0.11  0.27  0.47  0.82  1.00  2000
p3[3,1] 0.10  0.09  0.00  0.03  0.08  0.14  0.35  1.00  3000
p3[3,2] 0.54  0.18  0.20  0.41  0.54  0.67  0.86  1.00  2000
p3[3,3] 0.45  0.19  0.11  0.30  0.44  0.58  0.82  1.01  600
p3[3,4] 0.23  0.19  0.01  0.08  0.19  0.33  0.69  1.00  3000
p3[3,5] 0.32  0.23  0.01  0.13  0.28  0.48  0.83  1.00  3000
r     0.58  0.17  0.27  0.47  0.59  0.70  0.91  1.00  3000
psi   0.40  0.10  0.22  0.33  0.40  0.47  0.62  1.00  3000
n.occ[1] 24.13 2.94 17.00 22.00 25.00 26.00 29.00 1.00 1100
n.occ[2] 6.33 2.48 1.00 5.00 6.00 8.00 11.03 1.00 3000
n.occ[3] 9.54 2.90 5.00 7.00 9.00 11.00 16.00 1.00 1900
```

Many parameters are estimated with little precision, but we see that occupancy (`psi`) and the conditional (on occupancy) probability of successful reproduction (`r`) are estimated at higher values under model 2 than under model 1. We could also specify a model with covariate effects (Julian date in our data set) on these time-dependent parameters, but leave this for the exercises.

The multistate occupancy model can be extended in two important ways. First, the generalization to more than two occupancy states is straightforward. Second, a dynamic multistate occupancy model has been developed recently (MacKenzie et al., 2009). Similar to the multistate models of Chapter 9, these models estimate state transition probabilities. Technically, the state transition is an element of the state equation and can be included in WinBUGS by using a categorical distribution. The parameters of the state transition matrix may then be, for example, the probability that a site with reproduction in year  $t$  is abandoned in year  $t+1$ , or the probability that a site without reproduction in year  $t$  produces young in year  $t+1$ . Dynamic multistate occupancy models are conceptually analogous to multievent models (Pradel, 2005).

## 13.7 SUMMARY AND OUTLOOK

We have introduced site-occupancy models, a class of hierarchical logistic regression model for occurrence data that jointly estimate detection probability to account for imperfect detection. Occurrence may be a proxy for the local metapopulation abundance, which is the focus of interest in the binomial mixture model of the previous chapter. Alternatively, occupancy may be the focus of interest such as in species distribution models, disease ecology, or metapopulation ecology. When detection of occupied sites (patches) is not perfect, the extent of occurrence of species will be underestimated and covariate relationships will be estimated with bias, regardless of whether there are patterns in detection probability or whether it is constant. Given suitable data (occurrence observations that are replicated in both space and time within a short period), occupancy probability can be estimated separately from detection probability, and covariate relationships with either parameter can be estimated, even when the same covariate is affecting both occurrence and detection. Knowing typical values of detection probability and how the latter varies with measurable covariates can be invaluable for the planning of surveys.

We have furthermore illustrated a dynamic, multisession version of a site-occupancy model (MacKenzie et al., 2003; Royle and Kéry, 2007), which is precisely a generalization of a classical metapopulation model for incidence, colonization, and extinction probability that accounts for imperfect detection; imperfect detection biases virtually all parameter estimates in classical metapopulation models unless corrected for. Static and especially dynamic site-occupancy models have increasingly been used to correct for variation in effort over long time scales when studying changes in species distributions from historic data (Altwegg et al., 2008; Moritz et al., 2008; Tingley and Beissinger, 2009; Tingley et al., 2009; Kéry et al., 2010b; van Strien et al., 2011). We have also illustrated another important generalization, the multistate site-occupancy model (Royle and Link, 2005; Nichols et al., 2007). These models allow one to simultaneously deal with detection error and state uncertainty and thus considerably extend the range of possible applications of this model class. For instance, Miller et al. (2011) use multistate occupancy models to deal with false-negative (detection) and false-positive (misclassification) errors in occupancy data.

Further extensions of the basic model include Royle and Nichols (2003), who describe a heterogeneity site-occupancy model that allows one, under certain conditions, to estimate the mean abundance at a collection of sites from detection/nondetection data alone (see also Dorazio (2007); Conroy et al. (2008) for a Bayesian implementation). In an exciting new development, Bled et al. (2011b) describe complex, spatially explicit, dynamic

occupancy model for the spread of invasive species. Roth and Amrhein (2009) have developed a site-occupancy model to estimate local survival and recruitment from territory occupancy data with unmarked animals. Dorazio and Royle (2005) have described a multispecies site-occupancy model that enables one, among other things, to estimate species richness for each site (i.e., community size) as well as for the collection of sites (i.e., metacommunity size). The Bayesian implementation of this model using data augmentation (Dorazio et al., 2006) has been very seminal for community studies; see series of papers by Kéry and Royle (2008, 2009), Russell et al. (2009), Zipkin et al. (2009, 2010), and Ruiz-Gutiérrez and Zipkin (2011). This model has been extended to open population by Kéry et al. (2009; not including dynamics) and Dorazio et al. (2010; including occurrence dynamics); Yamaura et al. (2011) developed a version of the open multispecies site-occupancy model with the Royle-Nichols (2003) formulation of detection heterogeneity. In addition, MacKenzie et al. (2009) developed a multistate, dynamic occupancy model, which appears to be a very general and unifying model—most other occupancy models can be described as special cases of this overarching model. In summary, site-occupancy models represent an extremely powerful and flexible class of models for inference about populations of animals and plants.

## 13.8 EXERCISES

---

1. In the blue bug example, fit a “behavioral response” effect, that is, fit a separate detection probability dependent on whether the species has been detected ever before at a site or not. Hint, you can use the following R code to generate the “seen-before” covariate matrix. How do you interpret the results? Would you use the behavioral response model for inference about the system behind the blue bug data set? Discuss.

```
# Generate a 'seen-before' covariate
sb <- array(NA, dim = dim(y))
for (i in 1:27) {
  for (j in 1:6) {
    sb[i,j] <- max(y[i, 1:(j-1)])
  }
}
sb[is.na(y)] <- 0                                # Impute 'irrelevant' zeroes
```

2. In the dynamic occupancy model of Section 13.5.1, ignore the detection process and aggregate the temporal within-day replicates. Adapt the WinBUGS code to fit a conventional metapopulation model and see

how the estimated quantities are biased; see also Ruiz-Gutiérrez and Zipkin (2011).

3. Fit a multiseason, nondynamic version of the site-occupancy model to the burnet data. That is, treat days as a group and model occupancy independent between successive days (similar to how we modeled abundance in Section 12.3). In this way, you commit some pseudoreplication, but treating days as a group allows you to model occupancy as a function of temporally varying covariates.
4. Site-occupancy models represent the only currently available species distribution modeling framework that can estimate true, rather than apparent distributions (Kéry et al., 2010a; Kéry, 2011b). However, modeling occurrence and observation jointly can be difficult in marginal data situations. Devise a simulation study, where you vary the number of sites, occupancy, and detection probability as well as the number of replicate visits per site to see that in small-data situations, occupancy estimates will be biased high, and sometimes severely so. Do so in a model with constant detection and occurrence probability. Hint: this is a somewhat larger project.
5. In the multistate occupancy model, add an effect of Julian date on detection probability of hooting adults and begging young, that is,  $p_2$ ,  $p_{3,2}$  and  $p_{3,3}$ . Do not forget to standardize the covariate.

## 14

# Concluding Remarks

## OUTLINE

<b>14.1 The Power and Beauty of Hierarchical Models</b>	<b>464</b>
14.1.1 Hierarchical Models Make the Fitting of Complex Statistical Models Easier	464
14.1.2 Hierarchical Models Foster a Synthetic Understanding of a Large Array of Models	465
14.1.3 Hierarchical Models Lead to Cleaner Thinking	467
14.1.4 Hierarchical Models Lead to a Step-Up Approach in Tackling a Problem	467
14.1.5 What Kind of Hierarchical Model? Primary Model Selection in WinBUGS	468
14.1.6 Secondary Model Selection: Hierarchical Models and Variable Selection	469
14.1.7 Hierarchical Models and MARK, unmarked, E-SURGE, and PRESENCE	470
14.1.8 Hierarchical Models and Study Design	471
<b>14.2 The Importance of the Observation Process</b>	<b>472</b>
<b>14.3 Where Will We Go?</b>	<b>474</b>
14.3.1 Combination of Information	474
14.3.2 Population and Community Models for Metapopulation Designs	474
14.3.3 Spatial Models	475
14.3.4 Relaxing the Closure Assumption	475
14.3.5 More Flexible Covariate Modeling	476
14.3.6 Accounting for Misclassification Error	476
<b>14.4 The Importance of Population Analysis for Conservation and Management</b>	<b>476</b>

We have given an overview of the Bayesian analysis of models for population analysis using WinBUGS. By population analysis, we mean the analysis of data from populations and communities, especially data on distribution and abundance, and the four vital rates governing their dynamics: survival, recruitment, immigration, and emigration probability. We almost always use hierarchical models, which are fitted very naturally in a Bayesian framework of inference using the simple and flexible BUGS language. In particular, hierarchical models enable direct modeling of the observation process, thereby accounting for false-negative observation errors. We have extensively used capture–recapture-type models, which achieve a partitioning of an observed response into one component describing the ecological process of interest and another representing its imperfect observation. In this concluding chapter, we first reflect on the power and beauty of hierarchical models (Section 14.1) and on the importance of accounting for the observation process in any inference about populations and communities from ecological field data (Section 14.2). We continue with remarks on possible future avenues for population analysis (Section 14.3) and finally emphasize some of the applications in which rigorous population analyses appear to be particularly important (Section 14.4).

## 14.1 THE POWER AND BEAUTY OF HIERARCHICAL MODELS

The concept of hierarchical models runs as a common thread throughout this book. In this respect, we owe a great debt to Royle and Dorazio (2008), who have laid much of the intellectual groundwork for us and from which many code examples are taken. As we have seen many times in our book, the use of hierarchical models, especially when fitted in WinBUGS, has many advantages; some of which we discuss in the following.

### 14.1.1 Hierarchical Models Make the Fitting of Complex Statistical Models Easier

Hierarchical models express the observed data as a result of a sequence of linked, simpler probabilistic systems, with each random variable being dependent on the outcome of the random variable preceding it in the hierarchy of a model. A neat example is that for the dynamic occupancy model (Section 13.5). It consists of three linked Bernoulli random variables, or logistic regressions, describing the initial occupancy state,

occupancy dynamics, and the observation process, respectively. Ignoring site indices, this model can be written as follows:

Random variable 1 (initial state):  $z_1 \sim \text{Bernoulli}(\psi_1)$

Random variable 2 (state dynamics):  $z_{t+1} | z_t \sim \text{Bernoulli}(z_t \phi_t + (1 - z_t) \gamma_t)$

Random variable 3 (observation process):  $y_t | z_t \sim \text{Bernoulli}(z_t p_t)$

The hierarchical model decomposes a complicated stochastic system into a sequence of three dependent subprocesses. In this way, a complex model becomes much easier to understand and fit. The beauty of hierarchical models lies in the almost unbelievable simplicity when a complex model is decomposed into its individual component models. The entire model may be difficult to understand and fit for an ecologist, but each individual submodel is really simple. For example, the incorporation of additional complexity, such as covariate or random effects, is straightforward.

### 14.1.2 Hierarchical Models Foster a Synthetic Understanding of a Large Array of Models

A big advantage of the hierarchical specification of complex models in population analysis is that it fosters a whole new and synthetic understanding of a vast array of models, which may often be thought of as being unrelated. For instance, in the capture–recapture literature a big divide is typically made between models for closed and models for open populations. However, when written as a hierarchical model, we cannot avoid recognizing the similarities rather than seeing the differences between them. For instance, Table 14.1a shows the hierarchical specification of a closed-population model for abundance estimation implemented by data augmentation (Chapter 6) and that for an open population (dynamic occupancy model, Section 13.5).

In both the closed- and the open-population model, the initial state is modeled as a Bernoulli trial with a success parameter ( $\psi_1$ ). In the closed model,  $z_1 = 1$  denotes a genuinely existing individual, whereas in the open model  $z_1 = 1$  denotes an occupied site. The description of the state dynamics in the closed model is simply that there is none: the state at occasion  $t + 1$  is identical to the state at occasion  $t$ . In contrast, in the open model, the state at occasion  $t + 1$  is a function of the state at occasion  $t$  and parameters for survival ( $\phi$ ) and colonization ( $\gamma$ ). Finally, the observation process is again identical in both models: at time  $t$ , all existing individuals or occupied sites flip a coin to determine whether they are detected or not.

As another example, Table 14.1b shows the relationships between two superficially very different models, one an implicit and the other an explicit hierarchical model (Royle and Dorazio 2008). The first is a

**TABLE 14.1** Two examples of The Power of Hierarchical Models to Clarify the Relationships among Model Classes

(a) Conceptual Similarity between a Closed-Population Model (see Chapter 6) and an Open-Population Model (dynamic occupancy model; see Chapter 13)\*

Submodel	Closed-Population Model	Open-Population Model
Initial State	$z_1 \sim \text{Bernoulli}(\psi_1)$	$z_1 \sim \text{Bernoulli}(\psi_1)$
State Dynamics	$z_{t+1}   z_t = z_t$	$z_{t+1}   z_t \sim \text{Bernoulli}(z_t \phi_t + (1 - z_t) \gamma_t)$
Observation Process	$y_t   z_t \sim \text{Bernoulli}(z_t p_t)$	$y_t   z_t \sim \text{Bernoulli}(z_t p_t)$

(b) Conceptual Similarity between a State-Space Model (see Chapter 5) and the Cormack–Jolly–Seber Model (see Chapter 7)\*\*

Submodel	State-Space Model	CJS Model
Initial State	$N_1 \sim f(\theta_1)$	$z_1 = 1$
State Dynamics	$N_{t+1}   N_t \sim g(N_t, \theta_2)$	$z_{t+1}   z_t \sim g(z_t, \theta_2)$
Observation Process	$y_t   N_t \sim h(N_t, \theta_3)$	$y_t   z_t \sim h(z_t, \theta_3)$

\**t* indexes occasions in the closed-population model and primary occasions in the open-population model. To make all parameters in the latter identifiable, data are required from secondary occasions during a period of closure.

\*\*To clarify the difference between states in the two models, the relative abundance state at time *t* in the state-space model is denoted as  $N_t$ , whereas the “alive” state in the CJS model is denoted as  $z_t$ .

state-space model for a single time series of population counts (Chapter 5) and the second is the Cormack–Jolly–Seber (CJS) model for multiple time series of survival events (Chapter 7). The initial state is modeled using a distribution  $f$  with parameter  $\theta_1$  in the state-space model. In contrast, the initial state is not modeled in the CJS model but is 1 by definition because the CJS model conditions on initial capture. In both models, for all later time steps, the state at time  $t + 1$  is a random variable that depends on the state at time  $t$  and on parameter  $\theta_2$  via a distribution  $g$ . For  $g$ , we use a normal distribution with a mean and a variance parameter for the state-space model and a Bernoulli distribution with a parameter representing the success probability for the CJS model. Finally, as a description of the observation error at time  $t$ , we use a distribution  $h$ , which depends on the true state at time  $t$  and on parameter  $\theta_3$ . The statistical description of the observation process is another normal for the state space and a Bernoulli distribution for the CJS model. Similar schemes can be devised for all hierarchical models.

Hence, a hierarchical specification of models for population analysis emphasizes the similarities among models that appear to be very different at a first glance. We believe that the unified perspective on a large number of models provided by their hierarchical description will be very beneficial for your population modeling. Moreover, we have seen at many places how the specification of a model in the BUGS language is almost a direct

translation of a hierarchical model described in algebra, as in Table 14.1. Thus, we believe that statistical modeling in BUGS will foster in you a more synthetic understanding of large classes of models: they may *appear* quite different, but in fact they share many commonalities!

Throughout this book, we have emphasized the algebraic description of hierarchical models. There is another, very general way to describe hierarchical models: by directed acyclic graphs (DAGs; Spiegelhalter, 1998). We have not used DAGs in this book, except for Fig. 11.3, but if you have experience in reading and drawing DAGs, it can be very enlightening to draw them for the models in this book (R. A. Hutchinson, pers. comm.). This will also allow you to see the surprising similarities among these models.

### 14.1.3 Hierarchical Models Lead to Cleaner Thinking

We believe that hierarchical models foster a unified way of thinking about statistical modeling. For instance, for ecological field data, it becomes completely natural to think in terms of a hierarchical model that separates the ecological and the observation processes. Indeed, after a while of being in that mode of thinking, it becomes hard *not* to partition in your mind any kind of field data into the result of an ecological process and that of an observation process. So, indirectly, working with WinBUGS can be an act of intellectual hygiene that leads to a more mechanistic thinking about the systems we study.

### 14.1.4 Hierarchical Models Lead to a Step-Up Approach in Tackling a Problem

Although conceptually easy, fitting hierarchical models using WinBUGS can be complicated in practice and many things can go wrong. Hence, we always favor a modeling strategy where we start from a simple model and increasingly add in complexity until we are at the desired model (see Appendix 1, tip 28). This step-up approach may force on us a whole new way of thinking, where we start thinking about a problem in a very simplified way. Once we have understood the problem in that setting, we incrementally add in more complexity until we (hopefully) understand the model we wanted to fit to start with. We believe that this is exactly the way in which successful science should work: we must first understand a simple version of a problem before we can go on to try and understand the more complex ones. Often, we see people who immediately want to attack a very complex version of a problem, without first even trying to understand its simpler versions. We believe that this is a recipe for failure. Thus, modeling in WinBUGS trains us to think in a more disciplined way about doing science in general.

### 14.1.5 What Kind of Hierarchical Model? Primary Model Selection in WinBUGS

One of the aims of this book is to provide an accessible entry point to the literature on a large number of models that are useful for population analysis and to illustrate their implementation in WinBUGS. So which one in this bewildering multitude of models should we choose to answer our ecological questions? Obviously, the best choice among these models will depend on many things, including modeling objectives, data, sampling design and the assumptions one is willing to make.

The first criterion for primary model choice should always be our scientific or management questions, that is, the *objectives of the model*. This cannot be stressed enough: we ought to be very clear about the objective behind every modeling exercise. For example, prediction may require quite a different model than explanation. Further, model choice may depend on how similar the inferences (for instance, the estimate of the effect of a management intervention) under different models are and whether these differences are of practical importance given the modeling objectives. It may be reasonable to choose a second-best model when the inferences are practically identical to those under the best model and when the second-best model has other advantages (e.g., runs much faster).

Second, model choice depends on the data at hand. For instance, if data are available to estimate detection probability in a time series of counts, we would not use the rather simplistic description of the observation process of the models in Chapter 5. Rather, we might use a binomial or other suitable distribution to explicitly model false-negative detection errors (see Section 1.3). If no such information on detection probability is available, the implicit hierarchical models in Chapters 4 and 5 may be the best that can be done. Similarly, occurrence data such as territory occupancy or species distribution could be modeled using a variant of a binomial GLM (Chapters 3 and 4). Such an approach balls up in a single parameter the probabilities of occupancy and detection (Kéry et al. 2010a, Kéry 2011b). However, often we *do* have extra information on the observation process, for example, replicated detection or nondetection observations within a short time period. If this is the case, we find it hard to understand how one would choose *not* to jointly model the ecological process and the observation process in an explicit hierarchical model (i.e., a site-occupancy model, Chapter 13).

Sampling design is intimately linked to the previous point as are the assumptions that we may want or have to make. Often, we must make assumptions to compensate for deficiencies in the design and the data. For instance, absent data that are directly informative about the observation process, we must make strong assumptions about detection probability, for example, that it is constant on average or even that it is equal to 1.

Such primary decisions in the selection among broad classes of models require a great flexibility in our modeling and indeed an “organic” approach to statistical modeling, where the model can be tailored exactly to one’s needs. Fitting hierarchical models in WinBUGS typically gives one this flexibility.

### 14.1.6 Secondary Model Selection: Hierarchical Models and Variable Selection

Once we have made the primary decision about the general class of model for population analysis, there are additional decisions to take, for instance about the types or the form of covariates to incorporate. These secondary modeling decisions are often called model selection. Over the last decades, Burnham and Anderson (e.g., 2002) have been very influential in population ecology and management with their ideas about model building by a certain form of model selection. In a way, they advocate the opposite approach from the one that we are almost forced to adopt when using WinBUGS. Burnham and Anderson argue that, first, one must think hard about a problem and come up with a set of models that each represent a distinct biological hypothesis that we want to compare and test. Second, one must fit exactly these models to the data and use the Akaike’s information criterion (AIC) as a referee as to which model explains the observed data best.

We endorse this general strategy of doing science and like the emphasis on *a priori* thinking, instead of rushing into the data and doing data dredging. However, it appears to us that a strict adherence to the recipe of Burnham and Anderson is difficult with complicated models regardless of whether they are fitted in a frequentist or Bayesian way. As described in Section 14.1.4, there is much heuristic value in an incremental model building strategy. Thus, with WinBUGS, we are almost forced to do something that *resembles* data dredging, in which our fitting of a desired ultimate model often means that we have to fit several similar neighboring models as well. We believe that the ideal of Burnham and Anderson could still be upheld when a set of different *a priori* hypotheses are formulated before the start of the modeling exercise.

When teaching WinBUGS workshops to ecologists, one of the biggest disappointments we see is always the lack of an automated variable selection strategy that can be as easily implemented as AIC in maximum likelihood analyses. In hierarchical models, use of the standard DIC (Spiegelhalter et al., 2002) is controversial (Celeux et al., 2006) or downright wrong for at least some hierarchical models (Millar, 2009)—and most interesting models in this book are hierarchical. Furthermore, Bayesian variable selection is more involved and computationally demanding (O’Hara and Sillanpää, 2009).

The lack of an automated variable selection procedure can be seen as a good or as a bad thing. Of course, we would like to be able to filter through a large number of hierarchical models and have an easily computed criterion to help us pick the most useful one. On the other hand, we believe that the very ease with which AIC does variable selection has defeated part of the intentions of its advocates, Burnham and Anderson: since model comparison is made so easy, AIC may often lead to more rather than less data dredging. So, it may be that only when variable selection is difficult, do we actually think hard about which models we want to fit!

#### 14.1.7 Hierarchical Models and MARK, unmarked, E-SURGE, and PRESENCE

Hierarchical models can be fitted using frequentist and Bayesian methods. We believe that the choice between a frequentist and a Bayesian analysis of a model should in a large part be made on the basis of how practical it is and how well each one meets the objectives of the modeling. In this sense, we argue for a pragmatic choice between frequentist and Bayesian approaches (Little, 2006; Gelman and Hill, 2007; Gelman, 2008). Our own preference for Bayesian modeling using WinBUGS is, first and foremost, due to the fact that the BUGS language and the automated generation of MCMC algorithms in WinBUGS (and OpenBUGS or JAGS) have given us such a remarkable modeling freedom, one we have never experienced with any other software.

So, what about other frequentist software to fit population models as illustrated in this book? Examples of such software include MARK (White and Burnham, 1999), the new R package unmarked (Fiske and Chandler, 2011), E-SURGE (Choquet et al., 2009b), and PRESENCE (Hines, 2006). These useful software programs allow you to fit a very large number of models, many of them hierarchical, at least in concept (see Section 2.8). For instance, somebody commented to us that software MARK, along with its free and constantly evolving, excellent online manual of E. Cooch and G. White ([www.phidot.org/software/mark/docs/book](http://www.phidot.org/software/mark/docs/book)), is a competitor with our book or the way we do population modeling. We are not convinced that this is true. MARK, as well as unmarked, E-SURGE, and PRESENCE, represent tremendous endeavors that have done and continue to do an immeasurable service to the community of population ecologists to enhance the level of analyses that are possible and are conducted. For instance, MARK has about 100 different kinds of models that can be fitted by click and point techniques, and it has a reasonably unified layout of data entry and model building.

We think that MARK, unmarked, E-SURGE and PRESENCE and the modeling advocated in our book serve different audiences and targets:

for first-timers in population modeling or for someone who wants to fit some sort of standard model (which may be very sophisticated!), MARK or one of these other packages surely represent an ideal tool. By contrast, for users who want to fit nonstandard models, combine different models (integration of information) or use prior information, WinBUGS is the appropriate choice.

Finally, WinBUGS gives you a new way of thinking about statistical modeling in population analysis that comes with the model specification in the BUGS language. WinBUGS frees the modeler in you and allows for a fully “organic way” of model building. In WinBUGS, quick and easy jumps can often be made to totally different model classes by simply adding a line of code or two. In contrast, in software such as MARK, the same modification to a model might require one to start the whole modeling project anew, right from reading in the data into the software. Thereby, it is easy to lose sight of the underlying similarities of many of the models (as illustrated, for instance, in [Table 14.1](#)).

#### 14.1.8 Hierarchical Models and Study Design

Study design is an important topic that we have not covered in this book. We heard the comment that we ought to include in our book a chapter on study design. We first agreed, but in the end did not do this. The reason was simply that we thought the topic would be too daunting. We thought that the kinds of designs underlying the data featured in this book are too diverse and that we could not have said anything useful within a single book chapter. Instead, we refer to useful (non-Bayesian) books by Borchers et al. (2001), Buckland et al. (2001), Thompson (2002), and Williams et al (2002).

We would never want to downplay the importance of design in population ecology and wildlife management—actually, design is very important, and important first principles of good study design are almost always violated in ecological field studies, for example, random sampling. Nevertheless, with the great modeling freedom given by modern software such as WinBUGS, study design is perhaps a *little* less important nowadays than it was for the older, more rigid ways of analyzing data, for example, using ANOVA for designed experiments. The new modeling freedom brings with it a certain design freedom because even quite nonstandard designs can still be rigorously accommodated in a nonstandard analysis. Modeling in WinBUGS is perfect for nonstandard analyses (see also the Foreword by J. D. Nichols in Kéry, 2010).

One important design that we have only touched upon very superficially (see Section 10.8) is the robust design of sampling capture–recapture data (Pollock, 1982; Kendall et al., 1997; Williams et al., 2002). Except for

demographic analyses of abundance and occurrence data from metapopulation designs (Chapters 12 and 13), we have not illustrated this powerful design in this book. The robust design can be used also for demographic analyses of single populations to get improved estimates of population size and population dynamics (e.g., Karanth et al., 2006; Link and Barker, 2010). Often, data can be collected in the robust design with little added costs in the field. The robust design should be adopted whenever possible to improve inference in population analysis.

## 14.2 THE IMPORTANCE OF THE OBSERVATION PROCESS

---

We have emphasized a hierarchical perspective on models for population analysis. This is a reflection of the hierarchical processes that generate the observed data: an ecological process, which is usually of primary interest, and a dependent observation process, which often is a mere nuisance, but must be modeled to avoid spurious inferences about the ecological process. We have also emphasized what we believe is an important conceptual distinction between explicit and implicit hierarchical models (Royle and Dorazio, 2008); the former have parameters with an explicit ecological meaning and the latter do not. Typically, the distinction between explicit and implicit hierarchical models boils down to the question of whether our model contains an explicit description of the observation process, often in the form of a binomial distribution for a response representing a count or a Bernoulli for a response representing an event.

The direct interpretation of observed data, without accounting for the observation process, can be seriously misleading, even with constant detection probability. For instance, in Section 1.3, we saw that the average count seriously underestimated the true population size of sparrows in our yard. Moreover, repeated counts suggested that population size varies, whereas in reality, the variation in the counts was merely a consequence of the chance element inherent in any count based on an observation process characterized by imperfect detection. As another example, in Section 13.2, we saw how inferences from species distribution models can be quite wrong when imperfect detection is not accounted for. These difficulties may be aggravated severely when detectability is not constant but varies in response to environmental variables (see Sections 12.2.2 and 13.3.2).

But do we really always need to model the observation process in population analyses? After all, this usually requires additional data and more complex models. For instance, we need replicated counts for inference about abundance under an binomial mixture model (Chapter 12), whereas unreplicated counts suffice to model relative abundance under a simple Poisson regression (Chapter 3) or state-space model (Chapter 5).

We believe that explicit modeling of the observation process may not always be required or even possible. Most importantly, whether our models need a component for the observation process or not depends on the goals of our modeling exercise as well as on the available data. If we are satisfied with detecting *patterns* in abundance, occurrence, or survival, for instance, and courageous enough to assume that detection probability is constant over the desired dimensions of comparison (e.g., time), then we may well forego modeling of detection probability. Alternatively, we may have measured covariates that explain some or much of the variation in detection probability, and their inclusion in the model may standardize detection probability analytically (Link and Sauer, 2002; Sauer and Link, 2002). This may often be satisfactory. However, as soon as we want to interpret our data as true distribution, abundance, or survival, as opposed to relative distribution, relative abundance, and return rates, we must model detection probability explicitly and then also need the required data to do this (Kéry et al., 2010a). There is no escaping that. Treating a response in a model as relative abundance (i.e., not estimating detection probability as part of the model) and then reporting the results as “abundance” or “population size” is misleading and can be downright dishonest. When the required data to model detection explicitly are lacking (e.g., there are no replicated counts or no individual identification), then the model chosen should be as realistic as possible, but it must always be kept in mind that the model parameters confound the ecological and the observation process. The results of such an analysis must then be reported with due caution.

Perhaps, one ought to take a model-selection view of detection probability. Its inclusion in a model is associated with a cost as we need more and different data and a more complex model, so we ought to include it in the model only when its inclusion is warranted or practically feasible from a design standpoint. However, at the start, any rigorous population analysis ought to consider inclusion of detection probability as an essential model component. Only when we have reasons to exclude it should this be done. In other words, we believe that the choice of whether or not to include in a model a component for detection probability should be an active and conscious one and must be described and justified.

These considerations are also important at the design state of a study. We think that it should become much more natural to choose a design that includes collection of data informative on the observation process. To the extent possible, each ecological field study should be as follows:

1. Try to maximize detection probability and make it stable (eliminate as much variation in detection probability as possible, for instance, among observers, sites, and over time). Responses will still be variable (see Section 1.3), but at least some noise is eliminated.

2. Record the values of covariates that could be informative about the observation process, for instance, the experience and age of an observer, wind speed, and other measures of the conditions during a survey, and adjust your response variable for the effects of these nuisance variables.
3. Choose a design that enables explicit hierarchical models to be fitted, that is, collect data that allow detection probability to be estimated.

## 14.3 WHERE WILL WE GO?

In our book, we have presented a broad array of models useful for population analysis, but obviously, it was not possible to include everything. Population analysis is an extremely active field at the interface of ecology and statistics, so many new developments are foreseeable in the near future. Here, we highlight some new developments which we think are particularly relevant.

### 14.3.1 Combination of Information

The issue of combining different kinds of information in a single model is very general and includes the combined analysis of capture–recapture and mark-recovery data (Section 9.5) and integrated population models (Chapter 11). Such combinations represent the most efficient use of all available data from a study system and result in improved precision and increased number of parameters that can be estimated. This is especially important for rare and elusive species, where the increase from an effectively very small to a moderate amount of information may yield great benefits (Schaub et al., 2007; Kéry et al., 2011). Bayesian modeling is ideally suited for making such integrated modeling available to ecologists. We are likely to see increasing numbers of studies that combine all available data into a single model.

### 14.3.2 Population and Community Models for Metapopulation Designs

The development of site-occupancy models (MacKenzie et al. 2002; Tyre et al., 2003; Chapter 13) has started a rush in the development of models for the analysis of population and community data from metapopulation designs—collections of sites that are surveyed repeatedly. Models have been described for different kinds of responses (e.g., detection or nondetection data, counts), diverse ecological states (e.g., distribution, abundance, species richness), and static and dynamic systems. Metapopulation designs are extremely common especially in biodiversity

monitoring, but are frequently adopted also in ecological studies, and many new developments are likely for this kind of models in the near future. One example of a novel idea that combines metapopulation designs and the issue of combining information is an integrated metapopulation model (Conroy et al., 2008). Their model merges in a single analysis expensive data that directly inform on abundance (capture–recapture data), with cheaper data (detection/nondetection data), which can be collected over large areas for cheaper money.

### 14.3.3 Spatial Models

Spatial models describe the spatial dependence in a response or parameters due to neighborhood relations. Recent developments in statistics and computing increasingly allow modeling spatial relationships explicitly, rather than assuming them away or perhaps trying to eliminate them by careful study design. One example is occupancy modeling, where Royle and Dorazio (2008), Hines et al. (2010), and Bled et al. (2011) have described models where the occurrence of a species in one pixel, and possibly the associated colonization and extinction probabilities, depends on the state of neighboring pixels. This appears like an important and very obvious way of taking occupancy modeling to the next level of realism and ecological relevance. The same can also be undertaken for other types of spatially indexed data, for instance, for modeling abundance (Royle et al., 2007; Webster et al., 2008; Post van der Burg et al., 2011) and survival (Royle and Dubovský, 2001; Saracco et al., 2010). We are certain to see a great increase of this type of models in the future.

One special case of spatial models is spatially explicit capture–recapture models (Efford, 2004; Borchers and Efford, 2008; Royle and Young, 2008; see also Section 6.5). Essentially, all individual capture–recapture data are spatially indexed. Explicitly taking into account this spatial information allows to obtain less-biased estimators for quantities such as density or survival (Gardner et al., 2010) and to directly estimate other interesting quantities, such as the radius of a home-range or a dispersal kernel. This is a very active field of research with much to hope for in the near future.

### 14.3.4 Relaxing the Closure Assumption

Many ecological models of the capture–recapture type rely on the closure assumption: they assume that replicated observations can be made over some short time interval, when the system state is approximately constant. Obviously, no population in the field is ever entirely constant, so this assumption can only ever hold approximately. Violation of the closure assumption must be common and has various biasing effects on estimators of models that assume closure (Kendall, 1999; Rota et al., 2009).

Development of study designs and of models that do not need the closure assumption is therefore interesting. For counts from metapopulation designs, Dail and Madsen (2011) have developed a Jolly–Seber kind of binomial mixture model. Their model yields estimates of the population dynamics (abundance, survival, and recruitment) from temporally and spatially replicated counts that do not conform to the robust design (i.e., there is no replication for a site and year combination, say). This model is a great conceptual advantage and similar extensions may be possible in other situations.

#### 14.3.5 More Flexible Covariate Modeling

Modeling of nonlinear covariate relationships by using polynomials (e.g., Section 4.2) often gives sufficient flexibility. However, in many cases, more flexible covariate modeling is desirable, for instance, in the exploratory phase of an analysis, where one would like to “let the data speak for themselves”. Splines, general additive modeling and boosted regression trees have recently been developed within models illustrated in this book (Gimenez et al., 2006a,b; Collier et al., 2011; R. A. Hutchinson, personal communication). This is likely to become an ongoing development. Similarly, simpler parametric assumptions like the Poisson for the state in the binomial mixture models may be replaced with more flexible “nonparametric” assumptions (Dorazio et al., 2008).

#### 14.3.6 Accounting for Misclassification Error

A final topic we would like to highlight is the explicit modeling of misclassification. This had long had to be assumed away in capture–recapture models because simultaneous modeling of false-negative and false-positive errors proved too challenging. However, in recent years, there has been a surge of papers that deal with false-positive errors in addition to false-negative errors, for instance, in multistate models (Kendall et al., 2003; Pradel, 2005), for data from genetic analyses (Wright et al., 2009; Yoshizaki et al., 2009; Link et al., 2010) and in site-occupancy models (Royle and Link, 2006; Miller et al., 2011).

### 14.4 THE IMPORTANCE OF POPULATION ANALYSIS FOR CONSERVATION AND MANAGEMENT

In Chapter 1, we have claimed, perhaps with a little wink of our eyes, that for us the following equation is valid to a good approximation:

$$\text{Ecology} = \text{Population ecology} = \text{Population analysis}$$

The reason for this belief of ours is the central conceptual and practical importance of the population for all of ecology. In addition, population analysis is a pillar in many applied branches of ecology, such as wildlife and fisheries management and especially conservation biology. A rigorous scientific approach to conservation must be based on quantitative evidence and rely on the best available assessments of quantities such as distribution, abundance, species richness, population trend, and extinction probability or sustainability of a given harvest level (Caughley, 1994; Norris, 2004). These and other demographic quantities must be estimated, as well as possible, along with a full assessment in their uncertainty.

Unfortunately, the scientific standards of decision making in the arena of wildlife management and conservation in many countries are still extremely poor. Rather than basing decisions on hard evidence and rigorous science, unquantified claims and beliefs of stakeholders may be the sole basis for decisions. As an example, in 2011, the Swiss parliament approved of a change in the law that governs the control of predators of ungulates, such as wolf, bear, and lynx. It is now planned that complaints by one party of stakeholders, recreational hunters, about a decline in ungulate stocks be enough to trigger culling of these predators. The implementation details of the new law have yet to be worked out, but it is very likely that the decision to take action against these large predators will not be based on scientific standards of evidence, such as any scientifically defensible population analysis. In most Swiss cantons, there is not even a rigorous population monitoring of ungulates in place of the kind that is now commonplace for many taxa across Europe, including birds, plants, butterflies, and snails in Switzerland (Weber et al., 2004). Thus, even in countries such as Switzerland we still have a far way to go in making wildlife management evidence-based, rigorous, and scientifically defensible.

Recurring themes in population analyses for conservation biology are small sample sizes, data sets collected under nonstandard sampling designs or with no specified design at all, disparate data types, multiple levels of uncertainty, and stochasticity. As seen throughout this book, Bayesian population analysis using WinBUGS is extremely well suited for these challenges. Bayesian inference is exact for any sample size (Little, 2006). The remarkable modeling freedom given to the ecologist when using WinBUGS lets him or her adapt a model very flexibly to many idiosyncrasies of such data sets. The integration of information from disparate data types occurs very naturally (Chapter 11). Furthermore, multiple levels of uncertainty can be modeled flexibly using random effects. Thereby, all known components of uncertainty can be incorporated into an analysis and the combined uncertainty is propagated into all estimates and forecasts. Thus, we sincerely hope that Bayesian population analysis using WinBUGS not only makes you enjoy population modeling even more but also that it leads to better conclusions in science and to better decisions in wildlife management and conservation.

# A List of WinBUGS TRICKS

---

The appendix provides a list of tips that hopefully allow you to love WinBUGS more unconditionally. It is based on an appendix in the book *Introduction to WinBUGS for Ecologists* by Marc Kéry (2010), but also includes new stuff. We would suggest you skim over the list when you start working with WinBUGS and then refer back to it later as necessary.

1. Do read the manual: WinBUGS may not have the best documentation available for a software, but its manual is nevertheless very useful. Be sure to at least skim over most of it once when you start getting into WinBUGS, so you may remember that the manual has something to say about a particular topic when you need it. Do not forget the sections entitled “Tricks: Advanced Use of the BUGS Language” and “Tips and Troubleshooting”.
2. Always begin from a template: When starting a new analysis, *always* start from a template of a similar analysis. Only ever try to write an analysis from scratch if you want to test yourself.
3. Make a clear distinction between BUGS and R code: We use the R function `sink()` to write into the R working directory (which you can set yourself using `setwd()`) a text file containing the model description in the BUGS language. We find it practical to have all our codes in a single document. Here is a sketch example of how this looks like (you see this kind of code for each example where we use WinBUGS in the book).

```
# Define the model in the BUGS language
sink("modelname.txt")
cat("
model {           # BUGS model starts on this line
# Priors
some priors
```

```

# Likelihood
for (i in 1:n){
    some description of the likelihood
}
}                                # BUGS model ends with this line
",fill = TRUE)
sink()

```

Here, R code is left-aligned and BUGS code is indented one level and in bold. We neither show this indentation nor the bold type in the book, but show it here to clarify how everything inside the two quotes after “`cat`” and before “`,fill`” is BUGS code and everything outside these two quotes is R code. You have to be totally clear about which part of the code is in the BUGS language and which is in the R language. This may be a little confusing at first, especially because the two languages are quite similar: R is a dialect of S and BUGS is strongly inspired by S. Moreover, this way of writing the BUGS model sometimes seems to cause problems when using the R editor Tinn-R (see trick 17).

4. Give initial values: The wise choice of initial values can be the key to success or failure of an analysis in WinBUGS. With complex models, WinBUGS needs to start the Markov chains not too far away from their stationary distribution or it will crash or not even start to update. Of course, the requirement to start the chains close to the solution goes counter the requirement to start them at dispersed places in order to assess convergence, so some reasonable intermediate choice is important.
5. Do not select initial values that contradict the priors: Initial values must not be outside of the possible range of a parameter. For instance, negative initial values for a parameter that has prior mass only for positive values (such as a variance) will cause WinBUGS to crash and so do initial values outside of the range of a uniform prior.
6. Only provide initial values for quantities that appear in the model: Otherwise the “incompatible copy” error may appear.
7. Do not provide initial values for fixed elements of a vector-valued parameter: Sometimes some elements of a vector-valued parameter are known or fixed at a certain value. Then, they are no longer a parameter that is estimated and initial values must not be given for them. An example is a two-way fixed-effects ANOVA, where you must set to zero the effect of one level (e.g., the first) of one factor to avoid overparameterisation. In this case, no initial value is required for that effect. For a factor `beta` with four levels, the first of which is set to zero, you can do this as follows: `beta = c(NA, rnorm(3))`.
8. In your prior choice, be ignorant, but not too ignorant: When you want your Bayesian inference to be dominated by your data and choose

- priors intended to be vague, do not specify too much ignorance, otherwise traps may result or convergence may not be achieved. For instance, do not specify the limits of a uniform prior or the variance of a normal prior to be too wide.
9. How to deal with missing values (NAs): In WinBUGS, NAs are dealt with less automatically than in conventional stats programs with which you are likely familiar; hence, it is important to know how to deal with them: briefly, missing responses (i.e., missing  $y$ 's) are not a problem, but NAs in the explanatory variables (the  $x$ 's) need attention. A missing response is simply estimated, and indeed, adding missing responses for selected covariate values is one of the simplest ways to form predictions for desired values of explanatory variables (see Sections 5.4 and 11.5). On the other hand, a missing explanatory variable must either be replaced with some number, for example, the mean observed value for that variable, or else given some prior distribution. In general, the former is easier and should not pose a problem unless the number of missing  $x$ 's is large.
10. NAs and NaNs: When dealing with data in multidimensional arrays, a very useful R package is “reshape”. The newer versions of the reshape package in R 2.9 use an NA to fill in NAs. This makes WinBUGS very unhappy—you must have NA, not NaN. In general, this is probably good to know about BUGS, and newer versions of other packages may be doing the same thing. So, if you use the melt/cast functions in reshape to organize data, then you will need to update your code in the newer R versions by adding "fill=NA\_real\_". Example:  
`Ymat=cast(data.melt, SppCode~JulianDate~GridCellID,  
fun.aggregate=mean, fill=NA_real_)` (Beth Gardner, personal communication).
11. Data in arrays; think in a box (and know your box): When coding an analysis in WinBUGS, you often will have to deal with data that come in arrays, and these may have more than one dimension. For instance, when analyzing animal counts from different sites, over several years and taken at various months in a year, it may be useful to format them into a three-dimensional array. Some covariates of such an analysis will then have two or even three dimensions, too. You must then be absolutely clear about the dimensions of these “boxes” in which your data are and not get confused by the indexing of the data. In our experience, knowing how to format data into such arrays and then not getting lost is one of the most difficult things to learn about the routine use of WinBUGS.
12. Loop order in arrays: In “serious” analyses, your modeling will often require the data to be formatted in some multidimensional array. For instance, for a multispecies version of a site-occupancy model (Dorazio and Royle, 2005), you will have at least three dimensions

corresponding to species ( $i$ ), site ( $j$ ), and replicate survey ( $k$ ). It appears that how you build your array and, especially, how you loop over that array in the definition of the likelihood can make a huge difference in terms of the speed with which your Markov chains in WinBUGS evolve. You should loop over the longest dimension first and over the shortest last. For instance, if you have data from 450 sites, 100 species, and for two surveys each, then it appears best to format the data as  $y[j, i, k]$  and then loop over sites ( $j$ ) first, then over species ( $i$ ), and finally over replicate surveys ( $k$ ) (Beth Gardner and Elise Zipkin, personal communication).

13. Do not define things twice: Every parameter in WinBUGS can only be defined once. For instance, writing  $y \sim dnorm(mu, tau)$  and then adding  $y[3] <- 5$  will cause an error. There is a single exception to this rule, and that is the transformation of the response by some function such as the `log()` or `abs()`. So in order to conduct an analysis of a log-transformed response, you may write `log.y <- log(y)` and then `log.y ~ dnorm(mu, tau)`. Beware of inadvertently defining quantities multiple times when erroneously putting them within a loop that they do not belong.
14. Problems with WinBUGS' own logit function: We have sometimes experienced problems when using WinBUGS' own logit function, for instance, with achieving convergence. Therefore, it is often better to specify that transformation explicitly by `logit.p[i] <- log(p[i] / (1 - p[i]))`, `p[i] <- exp(logit.p[i]) / (1 + exp(logit.p[i]))` or `p[i] <- 1 / (1 + exp(-logit.p[i]))`.
15. "Stabilizing" the logit: To avoid numerical over- or underflow, you may "stabilize" the logit function by excluding extreme values (Brendan Wintle, personal communication). Here's a sketch of how to do that. The Gibbs sampling will typically get slower, but at least WinBUGS will be less likely to crash:

```
logit(psi.lim[i]) <- lpsi.lim[i]
lpsi.lim[i] <- min(999, max(-999, lpsi[i]))
lpsi[i] <- alpha.occ + beta.occ * something[i]
```

16. Truncated priors for normal random effects: Similarly, in log- or logit-normal mixtures (which we see when introducing a normal random effect into the linear predictor), you can truncate the zero-mean normal distribution, e.g., at  $\pm 20$  (Kéry and Royle, 2009):  $e[i] \sim dnorm(0, tau) I(-20, 20)$ . This can greatly help convergence of the Markov chains.
17. Problems with Tinn-R: Users of the popular R editor Tinn-R 2.0 (or newer) may have problems writing the text file containing the BUGS model description with the `sink()` function; Tinn-R adds to that file some gibberish that will cause WinBUGS to crash. You must then

use an alternative way of writing the model file. As an example, here is a workaround that should be compatible with Tinn-R (Wesley Hochachka, personal communication):

```
modelFilename = 'model.txt'  
cat ("  
model {  
# Here is the model in BUGS language  
}  
,fill=TRUE, file=modelFilename)
```

An alternative solution due to Jérôme Guélat is this: The “R send” functions available in Tinn-R allow sending commands into R. However, the “(echo=TRUE)” versions of these functions should not be used when sending the `sink()` function and its content into R. For example: one should use “R send: selection” instead of “R send: selection (echo=TRUE)”.

18. Run trial analyses first: Run very short chains first, for example, of length 12 with a burnin of 2, just to confirm that there are no coding or other errors. Only when you are satisfied that your code works and your model does what it should, increase the chain length to get a production run.
19. How to choose the burnin length: We have chosen Markov chain lengths so that convergence appears to be achieved. You may ask yourself how we decided on adequate chain lengths. The answer is simple: we always conduct trial runs first and based on that decide on the chain length for a production run of the analysis.
20. Avoid long Windows addresses: WinBUGS does not seem to like very long Windows addresses (C:\My harddisk\Important stuff\Less important stuff\ ...) for its working directory. Hence, you should not bury your WinBUGS analyses too far down in a tree hierarchy.
21. Use of native WinBUGS (1): A feature of both Kéry (2010) and this book is that WinBUGS is run exclusively from within program R. We believe that this is much more efficient than running native WinBUGS. However, with some complex models and/or large data sets, WinBUGS will be extremely slow. This may be the one exception where it is perhaps more efficient to run WinBUGS natively. You may still prepare the analysis in R as shown in this book, but only request WinBUGS to run very short Markov chains. When you set the option `DEBUG = TRUE` in the function `bugs()`, then WinBUGS will stay open after the requested number of iterations have been conducted. Then, you can request more iterations to be executed directly in WinBUGS (i.e., using the Update Tool; see chapter 4 in Kéry, 2010). You can then incrementally increase the total chain length and monitor convergence

- as you go. Once convergence has been achieved, do the required additional number of iterations and save them into coda files. You must do this latter, since when exiting WinBUGS, the `bugs()` function will only import back into R the (small) number of iterations that you originally requested. When you have your valuable samples of your complex model's posterior distribution in coda files, use facilities provided by R packages `boa` or `coda` to import them into R and process them (e.g., compute Brooks–Gelman–Rubin convergence tests or posterior summaries for inference about the parameters).
22. Use of native WinBUGS (2): Use of native WinBUGS can also be helpful to diagnose why a model does not run properly or produces unexpected results. If WinBUGS has been successfully called from R, there will be three (or more) text files in your current directory. If you have specified `working.directory = getwd()` in “`bugs`”, the files will be stored in the working directory (type `getwd()` if you are not sure which one this is). Otherwise, the files will be in a temporary directory (type `tempdir()` to see the path). The file stored first is the BUGS model and has the name that you have specified just after the `sink` command. The second file with the name `data.txt` contains the data in WinBUGS format. Finally, the third file contains the initial values in WinBUGS format and is named `inits1.txt`. If you have specified more than one chain, there will be more files of this type. To make use of native WinBUGS, you open WinBUGS, and select “file” → “new”. Then you copy the model text file, the data text file, and the text file(s) with the initial values into the empty window. You have then all information to start with a native WinBUGS analysis. See the WinBUGS documentation and chapter 4 in Kéry (2010) if you do not know how to proceed within WinBUGS.
23. Be flexible in your modeling: Try out different priors, for example, for parameters representing probabilities try a `uniform(0,1)`, a flat normal for the logit transform, or a `beta(1,1)`. Sometimes, one may work while another does not. Similarly, WinBUGS is very sensitive to changes in the parameterization of a model (see Gelman and Hill, 2007, for some good examples). Sometimes, one way of writing the model may work and the other does not, or one works much faster than the other.
24. Don't know how to specify the linear model? If you have trouble seeing how to specify a linear model in WinBUGS, use of the very handy R function `model.matrix()` may help. For instance, if you want to fit a model with four factors, A, B, C, and D, with all main effects and interactions A.B and C.D, do this to see how the linear model looks like: `model.matrix(~ A + B + C + D + A:B + C:D)`.
25. Scale continuous covariates: Scaling continuous covariates so that their range does not extend too far away on either side of zero, can greatly

- improve mixing of the chains and often makes convergence possible (see, e.g., Section 11.4. in Kéry, 2010, and Section 3.3.1 in this book).
- 26. What if WinBUGS hangs after compiling? Try a restart and if that does not work, find better starting values (this tip is from the manual).
  - 27. How to debug a WinBUGS analysis (1): If something went wrong, you need to attentively read through the entire WinBUGS log file from the top to identify the first thing that went wrong. Other errors may follow, but they may not be the actual cause of the failure.
  - 28. How to debug a WinBUGS analysis (2): When something does not work, the simplest and best advice (see also Gelman and Hill, 2007) is to go back to a simpler version of the same model, or to a similar model, that did work, and then incrementally increase the complexity of that model until you arrive at the desired model. That is, from less complex models *sneak up* on the model you want. Indeed, when using WinBUGS, you learn to always start from the simplest version of a problem and gradually build in more complexity until you are at the level of complexity that you require. We think that this is actually a very good approach to learning in general.
  - 29. DIC problems: Surprisingly, sometimes when getting a trap (including one with the very informative title “NIL dereference (read)”), setting the argument `DIC = FALSE` in the `bugs()` function has helped.
  - 30. What if R chokes on too much results from WinBUGS? Sometimes the R object created by `R2WinBUGS` is too big for one’s computer. Then, use `boa` or `coda` to read in the `coda` files directly and use their facilities to produce your inference in this way (e.g., convergence diagnostics, posterior summaries). Also see trick 37.
  - 31. Check of identifiability/estimability of parameters: To see whether two or more parameters are difficult to estimate separately, you can plot the values of their Markov chains against each other.
  - 32. Check of model adequacy: Do residual analysis, posterior predictive checks, and cross validation to see whether your model appears to be an adequate representation of the main features in the data.
  - 33. Predictions: The estimation of unobserved or future data is a very important part of inference. One particularly useful way to examine predictions is to estimate what a response would look like for a chosen combination of values of the explanatory variables. The generation and examination of such predicted values is an important method to understand complex models (for instance, to see what a particular interaction means) and also needed to illustrate the results of an analysis, for example, as a figure in a paper.
  - 34. Sensitivity analysis for priors: Consider assessing prior sensitivity, that is, repeat your analyses, or those for key models, with different prior specifications and see whether your inference is robust in this respect.

- If it is not, then not all is lost, but you must report on that in the methods section of your paper.
- 35. VISTA problems: Windows VISTA has caused all sorts of “challenges” in workshops taught—be prepared! One problem was that the default BUGS directory is not the same as that stated in the preface.
  - 36. Windows 7 problems: We experienced problems when WinBUGS is installed in a folder like “C:\Program Files\...”. WinBUGS runs fine when directly installed on “C:\”.
  - 37. Use of the coda package: Some prefer to work with coda objects than with the results returned by `bugs()` as we do throughout the book. Here is some R sample code to summarize the posterior samples and to check convergence and sample autocorrelation (from Richard B. Chandler):

```
outmc <- as.mcmc.list(out)
summary(outmc)
plot(outmc, ask=TRUE)
outmc[, "alpha"]
autocorr.plot(outmc, ask=TRUE)
gelman.plot(outmc, ask=TRUE)
gelman.diag(outmc)
window(outmc, thin=5)
```

- 38. Free choice among the sisters: When a model does not run in WinBUGS, you may try (and succeed) in OpenBUGS or JAGS, or vice versa (Mike Meredith, personal communication).
- 39. Beware of the dreaded “tiefschutz and probefahrt error” (Scott Sillett, Andy Royle, personal communication): It means that your computer has been invaded by space aliens. Stay calm, shut the door, leave the building, and set it on fire.
- 40. Last but not least, you must have a healthy distrust in your solutions: Always inspect your inference to see whether the WinBUGS solution makes sense with respect to what you know about the modeled system. For instance, look at tables of estimates and plot predictions against observed values for quantities that can be observed. Also watch out for unexplained differences in parameter estimates between neighboring models, for example, those that differ by only one covariate or some other rather minor model characteristic. This can be an indication that something went wrong (e.g. convergence was not reached or you made a coding error) or that there are estimability problems with the model for your data set.

# Two Further Useful Multistate Capture–Recapture Models

This appendix contains code for data simulation and analysis of two further multistate capture–recapture models.

## 2.1 ESTIMATION OF AGE-SPECIFIC SURVIVAL PROBABILITIES

### 2.1.1 Model Description

Age is one of the main predictors of survival in animal populations, and hence, estimation of age-specific survival is important in population analysis. Multistate models can be used to estimate age-specific survival; we define states as age classes. As a simple example, we consider the estimation of juvenile and adult survival. Juvenile survival refers to the interval from birth until the age of 1, and adult survival refers to all annual intervals after age 1. The states in this model are therefore “juvenile”, “adult”, and “dead”. Here is the state-transition matrix:

$$\begin{array}{ccccc} & & \text{juvenile} & \text{adult} & \text{dead} \\ \text{juvenile} & & 0 & \phi_{\text{juv}} & 1 - \phi_{\text{juv}} \\ \text{adult} & & 0 & \phi_{\text{ad}} & 1 - \phi_{\text{ad}} \\ \text{dead} & & 0 & 0 & 1 \end{array}$$

where  $\phi_{\text{juv}}$  is juvenile, and  $\phi_{\text{ad}}$  is adult apparent survival probability. As always, states are ordered from top down and from left to right in the same order as above. Juvenile survival is not on the diagonal of the transition matrix because juveniles become adult if they survive their first year of life.

The vector of observed states includes “seen as juvenile”, “seen as adult”, and “not seen”; hence, the observation matrix is

	seen as juvenile	seen as adult	not seen
juvenile	0	0	1
adult	0	$p$	$1-p$
dead	0	0	1

where  $p$  is recapture probability. The true states at time  $t$  correspond to the rows of the observation matrix in the order “alive as juvenile”, “alive as adult”, and “dead” from top down, and the observed states are in columns in the order “seen as juvenile”, “seen as adult”, and “not seen” from left to right. It may seem strange at first that juveniles cannot be seen—but if you think about it, the reason for it becomes clear juveniles may well be captured and marked, but the only possibility for them to be recaptured is in the following occasion at 1-year old and thus in the adult state.

### 2.1.2 Generation of Simulated Data

Let us assume that we marked young and adult little owls. Survival probability is 0.65 for adults and 0.3 for juveniles (from fledgling to their first anniversary), and recapture probability is 0.5 (note recapture refers only to adults). First, we simulate a data set, where juveniles are denoted as 1 and adults as 2.

```
# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phi.juv <- 0.3
phi.ad <- 0.65
p <- 0.5
n.occasions <- 6
n.states <- 3
n.obs <- 3
marked <- matrix(NA, ncol = n.states, nrow = n.occasions)
marked[,1] <- rep(200, n.occasions)      # Juveniles
marked[,2] <- rep(30, n.occasions)       # Adults
marked[,3] <- rep(0, n.occasions)         # Dead individuals

# Define matrices with survival, transition and recapture
# probabilities
# These are 4-dimensional matrices, with
# Dimension 1: state of departure
# Dimension 2: state of arrival
# Dimension 3: individual
# Dimension 4: time
# 1. State process matrix
totrel <- sum(marked)*(n.occasions-1)
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel,
n.occasions-1))
```

```

for (i in 1:totrel) {
  for (t in 1:(n.occasions-1)) {
    PSI.STATE[,,i,t] <- matrix(c(
      0, phi.juv, 1-phi.juv,
      0, phi.ad, 1-phi.ad,
      0, 0, 1), nrow = n.states, byrow = TRUE)
  } #t
} #i
# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel) {
  for (t in 1:(n.occasions-1)) {
    PSI.OBS[,,i,t] <- matrix(c(
      0, 0, 1,
      0, p, 1-p,
      0, 0, 1), nrow = n.states, byrow = TRUE)
  } #t
} #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked)
CH <- sim$CH

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed in WinBUGS!
# 1 = seen alive as juvenile, 2 = seen alive as adult, 3 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 3

```

We create only one data set, although we mark juveniles and adults. The capture-histories of individuals marked as juveniles start with a “1” and those of adults with a “2”. If we analyzed these data with a single-state capture–recapture (CJS) model, we would have to create an individual covariate containing the age (class) at first capture (see Section 7.7).

### 2.1.3 Analysis of the Model

```

# Specify model in BUGS language
sink("age.bug")
cat("
model {
# -----
# Parameters:
# phi.juv: juvenile survival probability
# phi.ad: adult survival probability
# p: recapture probability
# -----
# States (S):
# 1 alive as juvenile
# 2 alive as adult

```

```

# 3 dead
# Observations (O):
# 1 seen as juvenile
# 2 seen as adult
# 3 not seen
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi.juv[t] <- mean.phijuv
  phi.ad[t] <- mean.phiad
  p[t] <- mean.p
}
mean.phijuv ~ dunif(0, 1)
mean.phiad ~ dunif(0, 1)
mean.p ~ dunif(0, 1)

# Define state-transition and observation matrices
for (i in 1:nind){
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]: (n.occasions-1)){
    ps[1,i,t,1] <- 0
    ps[1,i,t,2] <- phi.juv[t]
    ps[1,i,t,3] <- 1-phi.juv[t]
    ps[2,i,t,1] <- 0
    ps[2,i,t,2] <- phi.ad[t]
    ps[2,i,t,3] <- 1-phi.ad[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- 0
    po[1,i,t,2] <- 0
    po[1,i,t,3] <- 1
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- p[t]
    po[2,i,t,3] <- 1-p[t]
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 0
    po[3,i,t,3] <- 1
  } #t
} #i

# State-space model likelihood
for (i in 1:nind){
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){
    # State equation: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation equation: draw O(t) given S(t)
    Y[i,t] ~ dcat(po[z[i,t], i, t-1,])
  } #t
} #i
}

```

```

  ", fill = TRUE)
sink()

# Bundle data
bugs.data <- list(Y = rCH, f = f, n.occasions = dim(rCH) [2], nind =
  dim(rCH) [1])

# Initial values
inits <- function() {list(mean.phijuv = runif(1, 0, 1), mean.phiad =
  runif(1, 0, 1), mean.p = runif(1, 0, 1), z = ch.init(rCH, f))}

# Parameters monitored
parameters <- c("mean.phijuv", "mean.phiad", "mean.p")

# MCMC settings
ni <- 2000
nt <- 3
nb <- 1000
nc <- 3

# Call WinBUGS from R (BRT 2 min)
age <- bugs(bugs.data, inits, parameters, "age.bug", n.chains = nc,
  n.thin = nt, n.iter = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())

```

The chains converge quite quickly.

```

print(age, digits = 3)

  mean    sd 2.5% 25% 50% 75% 97.5% Rhat n.eff
mean.phijuv 0.287 0.022 0.244 0.272 0.287 0.301 0.331 1.000 1000
mean.phiad  0.634 0.026 0.584 0.615 0.634 0.651 0.684 1.002 1000
mean.p      0.517 0.031 0.456 0.497 0.516 0.538 0.577 1.000 1000

```

The same model structure (age-dependent survival with two age classes) can be used to account for transients in the analyzed population and to estimate the probability that a newly caught individual is a transient (Pradel et al., 1997; Schaub et al., 2004b).

## 2.2 ACCOUNTING FOR IMMEDIATE TRAP RESPONSE

### 2.2.1 Model Description

One assumption of standard capture–recapture models in Chapters 7 and 9 is that all marked animals alive and available for capture at a given occasion have the same recapture probability. Sometimes this assumption is violated in a very specific way, namely that individuals captured at time  $t - 1$  have a different recapture probability at time  $t$  than individuals not captured at time  $t - 1$ . This is called trap response or behavioral response (see Sections 6.2.3 and 7.8). If recapture

probability at time  $t$  for individuals captured at  $t - 1$  is higher, trap response is called “trap happiness”, if recapture probability is lower, then it is called “trap shyness”. There are several mechanisms that may induce one or the other effect. Sometimes the effects are also induced by the sampling and do not reflect a behavioral change of the individuals. Nevertheless, trap response must be modeled; otherwise estimates for the other parameters will be biased. To account for immediate trap response, a multistate model can be used (Schaub et al., 2009). For a solution for the CJS model, see Section 7.8 and Pradel (1993b).

The states in this model are “alive and seen”, “alive and not seen” and “dead”. By this definition of the states, the observation process is included in the state-transition process. This is necessary, because the trap-response is a Markovian process which can be implemented in the state-transition matrix, but not in the observation matrix. The state-transition matrix is then

$$\begin{array}{ccccc} & \text{alive, seen} & \text{alive, not seen} & \text{dead} & \\ \text{alive, seen} & \left[ \begin{array}{ccc} \phi p_{ss} & \phi(1-p_{ss}) & 1-\phi \end{array} \right] & & & \\ \text{alive, not seen} & \left[ \begin{array}{ccc} \phi p_{ns} & \phi(1-p_{ns}) & 1-\phi \end{array} \right] & & & \\ \text{dead} & \left[ \begin{array}{ccc} 0 & 0 & 1 \end{array} \right] & & & \end{array}$$

Here,  $\phi$  is survival probability,  $p_{ss}$  is recapture probability if captured at the previous occasion (“seen”, “seen”), and  $p_{ns}$  is recapture probability if not captured at the previous occasion (“not seen”, “seen”).

The possible observations are “seen” and “not seen”. Because “seen” and “not seen” are defined and modeled the previous matrix, which formally describes the state process, assignment in the observation process is now deterministic. The observation transition matrix becomes

$$\begin{array}{ccccc} & \text{seen} & \text{not seen} & & \\ \text{alive, seen} & \left[ \begin{array}{cc} 1 & 0 \end{array} \right] & & & \\ \text{alive, not seen} & \left[ \begin{array}{cc} 0 & 1 \end{array} \right] & & & \\ \text{dead} & \left[ \begin{array}{cc} 0 & 1 \end{array} \right] & & & \end{array}$$

This model has identifiability problems. For example, when both recapture probabilities vary independently from each other over time, the model (often termed  $p_{t+m}$ ) is parameter-redundant (Gimenez et al., 2003). By contrast, models where both recapture probabilities are either linked with an additive time structure ( $p_{t+m}$ ) or are constant over time ( $p_m$ ) are not parameter redundant.

### 2.2.2 Generation of Simulated Data

We now generate the data as used in Section 7.8. Recall that we estimate survival of adult red-backed shrikes. We catch adults during the breeding season and mark them with color rings to facilitate

resighting and identification in the subsequent years. We survey all potential breeding territories each year. Typically we focus on breeding territories that were occupied by shrikes in previous years. If time allows, we search for other, newly established territories. Thus, marked individuals that survived and are philopatric to their territory have a higher chance of being resighted, while individuals that establish new territories are less likely to be found. However, once they are found, their chances of being resighted in the next year increases again. Such a sampling protocol induces a trap effect that biases survival unless accounted for. For data simulation, we assume survival  $\phi = 0.55$  and resighting probabilities  $p_{ss} = 0.75$  following a sighting in the preceding year and  $p_{ns} = 0.35$  if there was no sighting in the preceding occasion.

```

# Define mean survival, transitions, recapture, as well as number of
# occasions, states, observations and released individuals
phi <- 0.55
pss <- 0.75
pns <- 0.3
n.occasions <- 10
n.states <- 3
n.obs <- 2
marked <- matrix(NA, ncol = n.states, nrow = n.occasions)
marked[, 1] <- rep(100, n.occasions)      # Alive, seen
marked[, 2] <- rep(0, n.occasions)        # Alive, not seen
marked[, 3] <- rep(0, n.occasions)        # Dead

# Define matrices with survival, transition and recapture
probabilities
# These are 4-dimensional matrices, with
# Dimension 1: state of departure
# Dimension 2: state of arrival
# Dimension 3: individual
# Dimension 4: time
# 1. State process matrix
totrel <- sum(marked)*(n.occasions-1)
PSI.STATE <- array(NA, dim=c(n.states, n.states, totrel,
    n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.STATE[,,i,t] <- matrix(c(
      phi*pss, phi*(1-pss), 1-phi,
      phi*pns, phi*(1-pns), 1-phi,
      0,           0,           1       ), nrow = n.states, byrow =
      TRUE)
  } #t
} #i

# 2. Observation process matrix
PSI.OBS <- array(NA, dim=c(n.states, n.obs, totrel, n.occasions-1))
for (i in 1:totrel){
  for (t in 1:(n.occasions-1)){
    PSI.OBS[,,i,t] <- matrix(c(

```

```

1, 0,
0, 1,
0, 1 ), nrow = n.states, byrow = TRUE)
} #t
} #i

# Execute simulation function
sim <- simul.ms(PSI.STATE, PSI.OBS, marked, unobservable = 2)
CH <- sim$CH

# Compute vector with occasion of first capture
get.first <- function(x) min(which(x!=0))
f <- apply(CH, 1, get.first)

# Recode CH matrix: note, a 0 is not allowed in WinBUGS!
# 1 = seen, 2 = not seen
rCH <- CH # Recoded CH
rCH[rCH==0] <- 2

```

The capture–recapture data now consist only of 1 at occasions when an individual was seen and of a 2 at occasions when an individual was not seen.

### 2.2.3 Analysis of the Model

Again, to fit the trap response model in the BUGS code, we only alter the definition of matrices, priors, and possibly of the constraints.

```

# Specify model in BUGS language
sink("immtrap.bug")
cat("
model {
# -----
# Parameters:
# phi: survival probability
# pss: recapture probability at t, given captured at t-1
# pns: recapture probability at t, given not captured at t-1
# -----
# States (S):
# 1 alive, seen at t-1
# 2 alive, not seen at t-1
# 3 dead
# Observations (O):
# 1 seen
# 2 not seen
# -----
# Priors and constraints
for (t in 1:(n.occasions-1)){
  phi[t] <- mean.phi
  pss[t] <- mean.pss
  pns[t] <- mean.pns
}
mean.phi ~ dunif(0, 1)

```

```

mean.pss ~ dunif(0, 1)
mean.pns ~ dunif(0, 1)

# Define state-transition and observation matrices
for (i in 1:nind) {
  # Define probabilities of state S(t+1) given S(t)
  for (t in f[i]:(n.occasions-1)){
    ps[1,i,t,1] <- phi[t]*pss[t]
    ps[1,i,t,2] <- phi[t]*(1-pss[t])
    ps[1,i,t,3] <- 1-phi[t]
    ps[2,i,t,1] <- phi[t]*pns[t]
    ps[2,i,t,2] <- phi[t]*(1-pns[t])
    ps[2,i,t,3] <- 1-phi[t]
    ps[3,i,t,1] <- 0
    ps[3,i,t,2] <- 0
    ps[3,i,t,3] <- 1

    # Define probabilities of O(t) given S(t)
    po[1,i,t,1] <- 1
    po[1,i,t,2] <- 0
    po[2,i,t,1] <- 0
    po[2,i,t,2] <- 1
    po[3,i,t,1] <- 0
    po[3,i,t,2] <- 1
    } #t
  } #i
# State-space model likelihood
for (i in 1:nind) {
  z[i,f[i]] <- Y[i,f[i]]
  for (t in (f[i]+1):n.occasions){
    # State equation: draw S(t) given S(t-1)
    z[i,t] ~ dcat(ps[z[i,t-1], i, t-1,])
    # Observation equation: draw O(t) given S(t)
    Y[i,t] ~ dcat(po[z[i,t], i, t-1,])
    } #t
  } #i
}

",fill = TRUE)
sink()

# Bundle data
bugs.data <- list(Y = rCH, f = f, n.occasions = dim(rCH) [2], nind =
  dim(rCH) [1], z = known.state.ms(rCH, 2))

# Initial values
inits <- function(){list(mean.phi = runif(1, 0, 1), mean.pss =
  runif(1, 0, 1), mean.pns = runif(1, 0, 1), z = ms.init.z(rCH, f))}

# Parameters monitored
parameters <- c("mean.phi", "mean.pss", "mean.pns")

# MCMC settings
ni <- 20000
nt <- 6
nb <- 10000
nc <- 3

```

```
# Call WinBUGS from R (BRT 33 min)
immtrap <- bugs(bugs.data, inits, parameters, "immtrap.bug",
  n.chains = nc, n.thin = nt, n.ITER = ni, n.burnin = nb, debug = TRUE,
  bugs.directory = bugs.dir, working.directory = getwd())
print(immtrap, digits = 3)

      mean     sd   2.5%   25%   50%   75% 97.5%   Rhat n.eff
mean.phi 0.558 0.028 0.513 0.538 0.555 0.575 0.621 1.006 1000
mean.pss 0.776 0.040 0.692 0.750 0.779 0.805 0.847 1.006 660
mean.pns 0.312 0.097 0.145 0.239 0.307 0.376 0.517 1.006 790
```

Convergence is not obtained very quickly, so long MCMC runs are necessary. The use of information about the latent state variable  $z$  helps to obtain convergence more swiftly.

# References

---

- Abadi, F., Gimenez, O., Arlettaz, R., Schaub, M., 2010a. An assessment of integrated population models: bias, accuracy, and violation of the assumption of independence. *Ecology* 91, 7–14.
- Abadi, F., Gimenez, O., Ullrich, B., Arlettaz, R., Schaub, M., 2010b. Estimation of immigration rate using integrated population modeling. *J. Appl. Ecol.* 47, 393–400.
- Aebischer, A., 2009. *Der Rotmilan*. Haupt Verlag, Bern.
- Altwegg, R., Schaub, M., Roulin, A., 2007. Age-specific fitness components and their temporal variation in the barn owl. *Am. Nat.* 169, 47–61.
- Altwegg, R., Wheeler, M., Erni, B., 2008. Climate and the range dynamics of species with imperfect detection. *Biol. Lett.* 4, 581–584.
- Anderson, D.R., Burnham, K.P., White, G.C., 1985. Problems in estimating age-specific survival rates from recovery data of birds ringed as young. *J. Anim. Ecol.* 54, 89–98.
- Anderson, D.R., Burnham, K.P., White, G.C., 1994. AIC model selection in overdispersed capture-recapture data. *Ecology* 75, 1780–1793.
- Andrewartha, H.G., Birch, L.C., 1954. *The Distribution and Abundance of Animals*. University of Chicago Press, Chicago, IL.
- Arlettaz, R., Schaub, M., Fournier, J., Reichlin, T.S., Sierro, A., Watson, J.E.M., et al., 2010. From publications to public actions: when conservation biologists bridge the gap between research and implementation. *BioScience* 60, 835–842.
- Arnason, A.N., 1972. Parameter estimates from mark-recapture experiments on two populations subject to migration and death. *Res. Pop. Ecol.* 13, 97–113.
- Arnason, A.N., 1973. The estimation of population size, migration rates and survival in a stratified population. *Res. Pop. Ecol.* 15, 1–8.
- Arnason, A.N., Schwarz, C.J., 1999. Using POPAN-5 to analyse banding data. *Bird Study* 46, 157–168.
- Bailey, L.L., Converse, S.J., Kendall, W.L., 2010. Bias, precision and parameter redundancy in complex multistate models with unobservable states. *Ecology* 91, 1598–1604.
- Baillie, S.R., 1991. Integrated population monitoring of breeding birds in Britain and Ireland. *Ibis* 132, 151–166.
- Baillie, S.R., Brooks, S.P., King, R., Thomas, L., 2009. Using a state-space model of the British song thrush *Turdus philomelos* population to diagnose the causes of a population decline. Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 541–561.
- Balmford, A., Green, R.E., Jenkins, M., 2003. Measuring the changing state of nature. *Trend. Ecol. Evol.* 18, 326–330.
- Barker, R.J., 1997. Joint modeling of live-recapture, tag-resight, and tag-recovery data. *Biometrics* 53, 666–677.
- Barry, S.C., Brooks, S.P., Catchpole, E.A., Morgan, B.J.T., 2003. The analysis of ring-recovery data using random effects. *Biometrics* 59, 54–65.
- Bayes, T., 1763. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. R. Soc. A* 53, 370–418.
- Begon, M., Harper, J.L., Townsend, C.R., 1986. *Ecology: Individuals, Populations and Communities*. Blackwell, Oxford.

- Beissinger, S.R., 2002. Population viability analysis: past, present, future. In: Beissinger, S.R. (Ed.), *Population Viability Analysis*. The University of Chicago Press, Chicago, IL, pp. 5–17.
- Besbeas, P., Borysiewicz, R.S., Morgan, B.J.T., 2009. Completing the ecological jigsaw. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 513–539.
- Besbeas, P., Freeman, S.N., 2006. Methods for joint inference from panel survey and demographic data. *Ecology* 87, 1138–1145.
- Besbeas, P., Freeman, S.N., Morgan, B.J.T., 2005. The potential of integrated population modelling. *Aust. N. Z. J. Stat.* 47, 35–48.
- Besbeas, P., Freeman, S.N., Morgan, B.J.T., Catchpole, E.A., 2002. Integrating mark-recapture-recovery and census data to estimate animal abundance and demographic parameters. *Biometrics* 58, 540–547.
- Besbeas, P., Lebreton, J.D., Morgan, B.J.T., 2003. The efficient integration of abundance and demographic data. *App. Stat.* 52, 95–102.
- Bled, F., Royle, J.A., Cam, E., 2011a. Assessing hypotheses about nesting site occupancy dynamics. *Ecology* 92, 938–951.
- Bled, F., Royle, J.A., Cam, E., 2011b. Hierarchical modeling of an invasive spread: case of the Eurasian collared dove *Streptopelia decaocto* in the USA. *Ecol. Appl.* 21, 290–302.
- Bolker, B.M., 2008. *Ecological Models and Data in R*. Princeton University Press, Princeton, NJ.
- Bonner, S.J., Schwarz, C.J., 2006. An extension of the Cormack-Jolly-Seber model for continuous covariates with application to *Microtus pennsylvanicus*. *Biometrics* 62, 142–149.
- Borchers, D.L., Buckland, S.T., Zucchini, W., 2002. *Estimating Animal Abundance*. Springer, London.
- Borchers, D.L., Efford, M.G., 2008. Spatially explicit maximum likelihood methods for capture-recapture studies. *Biometrics* 64, 377–385.
- Borysiewicz, R.S., Morgan, B.J.T., Hénaux, V., Bregnalle, T., Lebreton, J.D., Gimenez, O., 2009. An integrated analysis of multisite recruitment, mark-recapture-recovery and multi-site census data. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 579–591.
- Boulinier, T., Nichols, J.D., Sauer, J.R., Hines, J.E., Pollock, K.H., 1998. Estimating species richness: the importance of heterogeneity in species detectability. *Ecology* 79, 1018–1028.
- Boyce, M.S., MacKenzie, D.I., Manly, B.F.J., Haroldson, M.A., Moody, D.S., 2001. Negative binomial models for abundance estimation of multiple closed populations. *J. Wildl. Manage.* 65, 498–509.
- Brooks, S.P., 2003. Bayesian computation: a statistical revolution. *Phil. Trans. R. Soc. A.* 361, 2681–2697.
- Brooks, S.P., Catchpole, E.A., Morgan, B.J.T., 2000a. Bayesian animal survival estimation. *Stat. Sci.* 15, 357–376.
- Brooks, S.P., Catchpole, E.A., Morgan, B.J.T., Barry, S.C., 2000b. On the Bayesian analysis of ring-recovery data. *Biometrics* 56, 951–956.
- Brooks, S.P., Catchpole, E.A., Morgan, B.J.T., Harris, M.P., 2002. Bayesian methods for analysing ringing data. *J. Appl. Stat.* 29, 187–206.
- Brooks, S.P., Gelman, A., 1998. Alternative methods for monitoring convergence of iterative simulations. *J. Comput. Graph. Stat.* 7, 434–455.
- Brooks, S.P., King, R., Morgan, B.J.T., 2004. A Bayesian approach to combining animal abundance and demographic data. *Anim. Biodiv. Cons.* 27.1, 515–529.
- Brown, J.H., Maurer, B.A., 1989. Macroecology: the division of food and space among species on continents. *Science* 243, 1145–1150.
- Brownie, C., Anderson, D.R., Burnham, K.P., Robson, D.S., 1985. *Statistical Inference from Band Recovery Data: A Handbook*. US Fish and Wildlife Service, Resource Publication 156, Washington, DC.

- Brownie, C., Hines, J.E., Nichols, J.D., 1986. Constant-parameter capture-recapture models. *Biometrics* 42, 561–574.
- Brownie, C., Hines, J.E., Nichols, J.D., Pollock, K.H., Hestbeck, J.B., 1993. Capture-recapture studies for multiple strata including non-Markovian transitions. *Biometrics* 49, 1173–1187.
- 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., Newman, K.B., Fernandez, C., Thomas, L., Harwood, J., 2007. Embedding population dynamics models in inference. *Stat. Sci.* 22, 44–58.
- Buckland, S.T., Newman, K.B., Thomas, L., Koesters, N.B., 2004. State-space models for the dynamics of wild animal populations. *Ecol. Mod.* 171, 157–175.
- Burnham, K.P., 1993. A theory for combined analysis of ring recovery and recapture data. In: Lebreton, J.D. (Ed), *Marked Individuals in the Study of Bird Populations*. Birkhäuser, Basel, pp. 199–213.
- Burnham, K.P., Anderson, D.R., 2002. *Model Selection and Multimodel Inference: A Practical Information Theoretic Approach*. Springer, New York.
- Burnham, K.P., Anderson, D.R., White, G.C., Brownie, C., Pollock, K.H., 1987. Design and analysis methods for fish survival experiments based on release-recapture. *Am. Fish. Soc. Monogr.* 5, 1–437.
- Burnham, K.P., White, G.C., 2002. Evaluation of some random effects methodology applicable to bird ringing data. *J. Appl. Stat.* 29, 245–264.
- Calvert, A.M., Bonner, S.J., Jonsen, I.D., Mills Flemming, J., Walde, S.J., Taylor, P.D., 2009. A hierarchical Bayesian approach to multi-state mark-recapture: simulations and applications. *J. Appl. Ecol.* 46, 610–620.
- Cam, E., Link, W.A., Cooch, E.G., Monnat, J.Y., Danchin, E., 2002. Individual covariation in life-history traits: seeing the trees despite the forest. *Am. Nat.* 159, 96–105.
- Carlin, B.P., Louis, T.A., 2009. *Bayesian Methods for Data Analysis*. CRC Press/Taylor & Francis Group, Boca Raton, FL.
- Caswell, H., 1988. Theory and models in ecology: a different perspective. *Ecol. Mod.* 43, 33–44.
- Caswell, H., 2001. *Matrix Population Models. Construction, Analysis, and Interpretation*. Sinauer Associates, Sunderland, MA.
- Catchpole, A.E., Morgan, B.J.T., Tavecchia, G., 2008. A new method for analysing discrete life history data with missing covariate values. *J. R. Stat. Soc. B* 70, 445–460.
- Catchpole, E.A., Kgosi, P.M., Morgan, B.J.T., 2001. On the near-singularity of models for animal recovery data. *Biometrics* 57, 720–726.
- Catchpole, E.A., Morgan, B.J.T., 1997. Detecting parameter redundancy. *Biometrika* 84, 187–196.
- Caughley, G., 1994. Directions in conservation biology. *J. Anim. Ecol.* 63, 215–244.
- Celeux, G., Forbes, F., Robert, C.P., Titterington, D.M., 2006. Deviance information criteria for missing data models. *Bayesian Anal.* 1, 651–674.
- Chandler, R.B., King, D.I., 2011. Golden-winged warbler habitat selection and habitat quality in Costa Rica: an application of hierarchical models for open populations. *J. Appl. Ecol.* 48, 1038–1047.
- Chandler, R.B., King, D.I., Chandler, C.C., 2009a. Effects of management regime on the abundance and nest survival of shrubland birds in wildlife openings in northern New England, USA. *Forest Ecol. Manage.* 258, 1669–1676.
- Chandler, R.B., King, D.I., DeStefano, S., 2009b. Scrub-shrub bird habitat associations at multiple spatial scales in beaver meadows in Massachusetts. *Auk* 126, 186–197.
- Choquet, R., Lebreton, J.D., Gimenez, O., Reboulet, A.M., Pradel, R., 2009a. U-CARE: utilities for performing goodness of fit tests and manipulating CApture-REcapture data. *Ecography* 32, 1071–1074.

- Choquet, R., Rouan, L., Pradel, R., 2009b. Program E-SURGE: a software application for fitting multievent models. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 845–865.
- Choquet, R., Reboulet, A.M., Pradel, R., Lebreton, J.D., 2001. U-CARE (Utilities-Capture-Recapture) User's Guide. CEFE/CNRS, Montpellier, France.
- Clark, J.S., 2005. Why environmental scientists are becoming Bayesians. *Ecol. Lett.* 8, 2–14.
- Clark, J.S., Björnstad, O.N., 2004. Population time series: process variability, observation errors, missing values, lags, and hidden states. *Ecology* 85, 3140–3150.
- Clark, J.S., Ferraz, G., Oguge, N., Hays, H., DiCostanzo, J., 2005. Hierarchical Bayes for structured, variable populations: from recapture data to life-history prediction. *Ecology* 86, 2232–2244.
- Clobert, J., Lebreton, J.D., 1991. Estimation of demographic parameters in bird populations. In: Perrins, C.M. (Ed.), *Bird Population Studies*. Oxford University Press, Oxford, pp. 75–104.
- Collier, B.A., Groce, J.E., Morrison, M.L., Newnam, J.C., Campomizzi, A.J., Farrell, S.L., et al., 2011. Predicting patch occupancy in fragmented landscapes at the rangewide scale for endangered species: an example of an American warbler. *Div. Dist.* (in press).
- Conn, P.B., Cooch, E.G., 2009. Multistate capture-recapture analysis under imperfect state observation: an application to disease models. *J. Appl. Ecol.* 46, 486–492.
- Conroy, M.J., Runge, J.P., Barker, R.J., Schofield, M.R., Fonnesbeck, C.J., 2008. Efficient estimation of abundance for patchily distributed populations via two-phase, adaptive sampling. *Ecology* 89, 3362–3370.
- Cormack, R.M., 1964. Estimates of survival from the sighting of marked animals. *Biometrika* 51, 429–438.
- Coull, B.A., Agresti, A., 1999. The use of mixed logit models to reflect heterogeneity in capture-recapture studies. *Biometrics* 55, 294–301.
- Crawley, M.J., 2005. *Statistics. An Introduction Using R*. Wiley, Chichester, West Sussex.
- Cressie, N., Calder, C.A., Clark, J.S., Ver Hoef, J.M., Wikle, C.K., 2009. Accounting for uncertainty in ecological analysis: the strengths and limitations of hierarchical statistical modeling. *Ecol. Appl.* 19, 553–570.
- Crosbie, S.F., Manly, B.F.J., 1985. Parsimonious modeling of capture-mark-recapture studies. *Biometrics* 41, 385–398.
- Dail, D., Madsen, L., 2011. Models for estimating abundance from repeated counts of an open population. *Biometrics* 67, 577–587.
- David, O., Garnier, A., Larédo, C., Lecomte, J., 2010. Estimation of plant demographic parameters from stage-structured censuses. *Biometrics* 66, 875–882.
- De Valpine, P., 2011. Frequentist analysis of hierarchical models for population dynamics and demographic data. *J. Ornithol.* (in press).
- De Valpine, P., Hastings, A., 2002. Fitting population models incorporating process noise and observation error. *Ecol. Monogr.* 72, 57–76.
- Dennis, B., 1996. Discussion: should ecologists become Bayesian? *Ecol. Appl.* 6, 1095–1103.
- Dennis, B., Munholland, P.L., Scott, J.M., 1991. Estimation of growth and extinction parameters for endangered species. *Ecol. Monogr.* 61, 115–143.
- Dennis, B., Ponciano, J.M., Lele, S.R., Taper, M.L., Staples, D.F., 2006. Estimating density dependence, process noise, and observation error. *Ecol. Monogr.* 76, 323–341.
- Dennis, B., Taper, M.L., 1994. Density dependence in time series observations of natural populations: estimation and testing. *Ecol. Monogr.* 64, 205–224.
- Dobson, A., Barnett, A., 2008. *An Introduction to Generalized Linear Models*. CRC/Chapmann & Hall, Boca Raton, FL.
- 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.
- Dorazio, R.M., Kéry, M., Royle, J.A., Plattner, M., 2010. Models for inference in dynamic metacommunity systems. *Ecology* 91, 2466–2475.
- Dorazio, R.M., Mukherjee, B., Zhang, L., Ghosh, M., Jelks, H.L., Jordan, F., 2008. Modeling unobserved sources of heterogeneity in animal abundance using a Dirichlet process prior. *Biometrics* 64, 635–644.
- Dorazio, R.M., Royle, J.A., 2003. Mixture models for estimating the size of a closed population when capture rates vary among individuals. *Biometrics* 59, 351–364.
- Dorazio, R.M., Royle, J.A., 2005. Estimating size and composition of biological communities by modeling the occurrence of species. *J. Am. Stat. Assoc.* 100, 389–398.
- Dorazio, R.M., Royle, J.A., Söderström, B., Glimskär, A., 2006. Estimating species richness and accumulation by modeling species occurrence and detectability. *Ecology* 87, 842–854.
- Dupuis, J.A., 1995. Bayesian estimation of movement and survival probabilities from capture-recapture data. *Biometrika* 82, 761–772.
- Dupuis, J.A., Schwarz, C.J., 2007. A Bayesian approach to the multistate Jolly-Seber capture-recapture model. *Biometrics* 63, 1015–1022.
- Duriez, O., Saether, S.A., Ens, B.J., Choquet, R., Pradel, R., Lambeck, R.H.D., et al., 2009. Estimating survival and movements using both live and dead recoveries: a case study of oystercatchers confronted with habitat change. *J. Appl. Ecol.* 46, 144–153.
- Efford, M., 2004. Density estimation in live-trapping studies. *Oikos* 106, 598–610.
- Efford, M.G., Borchers, D.L., Byrom, A.E., 2009a. Density estimation by spatially explicit capture-recapture: likelihood-based methods. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 255–269.
- Efford, M.G., Dawson, D.K., 2009. Effect of distance-related heterogeneity on population size estimates from point counts. *Auk* 126, 100–111.
- Efford, M.G., Dawson, D., Borchers, D.L., 2009b. Population density estimated from locations of individuals on a passive detector array. *Ecology* 90, 2676–2682.
- Elith, J., Leathwick, J.R., Hastie, T., 2008. A working guide to boosted regression trees. *J. Anim. Ecol.* 77, 802–813.
- Elliott, M.R., Little, R.J.A., 2000. A Bayesian approach to combining information from a census, a coverage measurement survey, and demographic analysis. *J. Am. Stat. Assoc.* 95, 351–362.
- Ellison, A.M., 2004. Bayesian inference in ecology. *Ecol. Lett.* 7, 509–520.
- Engen, S., Saether, B.E., Sverdrup-Thygeson, A., Grotan, V., Odegaard, F., 2008. Assessment of species richness from species abundance distributions at different localities. *Oikos* 117, 738–748.
- Fewster, R.M., Buckland, S.T., Siriwardena, G.M., Baillie, S.R., Wilson, J.D., 2000. Analysis of population trends for farmland birds using generalized additive models. *Ecology* 81, 1970–1984.
- Fiske, I.J., Chandler, R., 2011. Unmarked: An R package for the analysis of wildlife occurrence and abundance data. *J. Stat. Softw.* (in press).
- Fletcher, D., 1994. A mark-recapture model in which sighting probability depends on the number of sightings on the previous occasion. In: Fletcher, D., Manly, B.F.J. (Eds.), *Statistics in Ecology and Environmental Monitoring*. Otago University Press, Dunedin, pp. 105–110.
- Franklin, A.B., Anderson, D.R., Gutierrez, R.J., Burnham, K.P., 2000. Climate, habitat quality, and fitness in Northern spotted owl populations in Northwestern California. *Ecol. Monogr.* 70, 539–590.
- Freckleton, R.P., Watkinson, A.R., Green, R.E., Sutherland, B.J., 2006. Census error and the detection of density dependence. *J. Anim. Ecol.* 75, 837–851.

- Frederiksen, M., Bregnballe, T., 2000. Evidence for density-dependent survival in adult cormorants from a combined analysis of recoveries and resightings. *J. Anim. Ecol.* 69, 737–752.
- Gaillard, J.M., Viallefond, A., Loison, A., Festa-Bianchet, M., 2004. Assessing senescence patterns in populations of large mammals. *Anim. Biodiv. Cons.* 27.1, 47–58.
- Gardner, B., Reppucci, J., Lucherini, M., Royle, J.A., 2010. Spatially explicit inference for open populations: estimating demographic parameters from camera-trap studies. *Ecology* 91, 3376–3383.
- Gardner, B., Royle, J.A., Wegan, M.T., 2009. Hierarchical models for estimating density from DNA mark-recapture studies. *Ecology* 90, 1106–1115.
- Gaston, K.J., Blackburn, T.M., 2000. Pattern and Process in Macroecology. Blackwell Science, Oxford.
- Gauthier, G., Besbeas, P., Lebreton, J.D., Morgan, B.J.T., 2007. Population growth in snow geese: a modeling approach integrating demographic and survey information. *Ecology* 88, 1420–1429.
- Gauthier, G., Lebreton, J.D., 2008. Analysis of band-recovery data in a multistate capture-recapture framework. *Can. J. Stat.* 36, 59–73.
- Gelman, A., 2005. Analysis of variance: why is it more important than ever (with discussion). *Ann. Stat.* 33, 1–53.
- Gelman, A., 2006. Prior distributions for variance parameters in hierarchical models. *Bayesian Anal.* 1, 515–534.
- Gelman, A., 2008. Objections to Bayesian statistics (with discussion). *Bayesian Anal.* 3, 445–450.
- Gelman, A., Carlin, J.P., Stern, H.S., Rubin, D.B., 2004. Bayesian Data Analysis. CRC/Chapman & Hall, Boca Raton, FL.
- Gelman, A., Hill, J., 2007. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press, Cambridge.
- Gelman, A., Meng, X.-L., Stern, H.S., 1996. Posterior predictive assessment of model fitness via realized discrepancies (with discussion). *Stat. Sinica* 6, 733–807.
- Geman, S., Geman, D., 1984. Stochastic relaxion, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pat. Anal. Mach. Intell.* 6, 721–741.
- Gibbons, D.W., Donald, P.F., Bauer, H.G., Fornasari, L., Dawson, I.K., 2007. Mapping avian distributions: the evolution of bird atlases. *Bird Study* 54, 324–334.
- Gilks, W.R., Thomas, A., Spiegelhalter, D.J., 1994. A language and program for complex Bayesian modelling. *Statistician* 43, 169–177.
- Gimenez, O., Bonner, S.J., King, R., Parker, R.A., Brooks, S.P., Jamieson, L.E., et al., 2009a. WinBUGS for population ecologists: Bayesian modeling using Markov Chain Monte Carlo methods. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), Modeling Demographic Processes in Marked Populations. Springer, New York, pp. 883–915.
- Gimenez, O., Choquet, R., Lebreton, J.D., 2003. Parameter redundancy in multistate capture-recapture models. *Biomet. J.* 45, 704–722.
- Gimenez, O., Covas, R., Brown, C.R., Anderson, M.D., Bomberger Brown, M., Lenormand, T., 2006a. Nonparametric estimation of natural selection on a quantitative trait using mark-recapture data. *Evolution* 60, 460–466.
- Gimenez, O., Crainiceanu, C., Barbraud, C., Jenouvrier, S., Morgan, B.J.T., 2006b. Semiparametric regression in capture-recapture modeling. *Biometrics* 62, 691–698.
- Gimenez, O., Morgan, B.J.T., Brooks, S.P., 2009b. Weak identifiability in models for mark-recapture-recovery data. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), Modeling Demographic Processes in Marked Populations. Springer, New York, pp. 1055–1067.
- Gimenez, O., Rossi, V., Choquet, R., Dehais, C., Doris, B., Varella, H., et al., 2007. State-space modelling of data on marked individuals. *Ecol. Mod.* 206, 431–438.
- Gimenez, O., Viallefond, A., Catchpole, A.E., Choquet, R., Morgan, B.J.T., 2004. Methods for investigating parameter redundancy. *Anim. Biodiv. Cons.* 27.1, 561–572.

- Gimenez, O., Viallefont, A., Charmantier, A., Pradel, R., Cam, E., Brown, C.R., et al., 2008. The risk of flawed inference in evolutionary studies when detectability is less than one. *Am. Nat.* 172, 441–448.
- Gotelli, N.J., McGill, B.J., 2006. Null versus neutral models: what's the difference? *Ecography* 29, 793–800.
- Gould, W.R., Nichols, J.D., 1998. Estimation of temporal variability of survival in animal populations. *Ecology* 79, 2531–2538.
- Grosbois, V., Gimenez, O., Gaillard, J.M., Pradel, R., Barbraud, C., Clobert, J., et al., 2008. Assessing the impact of climate variation on survival in vertebrate populations. *Biol. Rev.* 83, 357–399.
- Grosbois, V., Harris, M.P., Anker-Nilssen, T., McCleery, R.H., Shaw, D.N., Morgan, B.J.T., et al., 2009. Modeling survival at multi-population scales using mark-recapture data. *Ecology* 90, 2922–2932.
- Guillera-Arroita, G., Ridout, M.S., Morgan, B.J.T., 2010. Design of occupancy studies with imperfect detection. *Meth. Ecol. Evol.* 1, 131–139.
- Guisan, A., Thuiller, W., 2005. Predicting species distribution: offering more than simple habitat models. *Ecol. Lett.* 8, 993–1009.
- Hagemeijer, W., Blair, M., 1998. The EBCC Atlas of European Breeding Birds. T. & A.D. Poyser, London.
- Hanski, I., 1994. A practical model for metapopulation dynamics. *J. Anim. Ecol.* 63, 151–162.
- Hanski, I., 1998. Metapopulation dynamics. *Nature* 396, 41–49.
- Hargrove, J.W., Borland, C.H., 1994. Pooled population parameter estimates from mark-recapture data. *Biometrics* 50, 1129–1141.
- Harper, J.L., 1977. Population Biology of Plants. Academic Press, London.
- Harvey, A.C., 1989. Forecasting, Structural Time Series Models and Kalman Filter. Cambridge University Press, Cambridge.
- Hastie, T.J., Tibshirani, R.J., 1990. Generalized Additive Models. Chapman & Hall/CRC, Boca Raton, FL.
- Hastings, W.K., 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 97–109.
- Hendriks, I.E., Deudero, S., Basso, L., Cabanellas-Reboredo, M., Alvarez, E., 2011. Growth rates of juvenile and adult *Pinna nobilis* around Majorca (Mediterranean, Spain). (in prep.).
- Hestbeck, J.B., Nichols, J.D., Malecki, R.A., 1991. Estimates of movement and site fidelity using mark-resight data of wintering Canada Geese. *Ecology* 72, 523–533.
- Hines, J.E., 2006. PRESENCE2: Software to estimate patch occupancy and related parameters. USGS-PWRC, Laurel, MD.
- Hines, J.E., Nichols, J.D., Royle, J.A., MacKenzie, D.I., Gopalaswamy, A.M., Samba Kumar, N., et al., 2010. Tigers on trails: occupancy modeling for cluster sampling. *Ecol. Appl.* 20, 1456–1466.
- Hooten, M.B., Wikle, C.K., Dorazio, R.M., Royle, J.A., 2007. Hierarchical spatiotemporal matrix models for characterizing invasions. *Biometrics* 63, 558–567.
- Hubbell, S.P., 2001. A Unified Theory of Biodiversity and Biogeography. Princeton University Press, Princeton, NJ.
- Huntley, B., Green, R.E., Collingham, Y.C., Willis, S.G., 2007. A Climatic Atlas of European Breeding Birds. Lynx Edicions, Barcelona.
- Hurlbert, S.H., 1984. Pseudoreplication and the design of ecological field experiments. *Ecol. Monogr.* 54, 187–211.
- Jetz, W., Rahbek, C., 2002. Geographic range size and the determinants of avian species richness. *Science* 297, 1548–1551.
- Joe, M., Pollock, K.H., 2002. Separation of survival and movement rates in multi-state tag-return and capture-recapture models. *J. Appl. Stat.* 29, 373–384.
- Jolly, G.M., 1965. Explicit estimates from capture-recapture data with both death and immigration-stochastic model. *Biometrika* 52, 225–247.

- Joseph, L.N., Elkin, C., Martin, T.G., Possingham, H., 2009. Modeling abundance using N-mixture models: the importance of considering ecological mechanisms. *Ecol. Appl.* 19, 631–642.
- Kadane, J.B., Lazar, N.A., 2004. Methods and criteria for model selection. *J. Am. Stat. Assoc.* 99, 279–290.
- Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *J. Basic Eng. D* 82, 35–45.
- Karanth, K.U., Nichols, J.D., Kumar, N.S., Hines, J.E., 2006. Assessing tiger population dynamics using photographic capture-recapture sampling. *Ecology* 87, 2925–2937.
- Kendall, W.L., 1999. Robustness of closed capture-recapture methods to violations of the closure assumption. *Ecology* 80, 2517–2525.
- Kendall, W.L., Conn, P.B., Hines, J.E., 2006. Combining multistate capture-recapture data with tag recoveries to estimate demographic parameters. *Ecology* 87, 169–177.
- Kendall, W.L., Hines, J.E., Nichols, J.D., 2003. Adjusting multistate capture-recapture models for misclassification bias: manatee breeding proportions. *Ecology* 84, 1058–1066.
- Kendall, W.L., Nichols, J.D., 2002. Estimating state-transition probabilities for unobservable states using capture-recapture/resighting data. *Ecology* 83, 3276–3284.
- Kendall, W.L., Nichols, J.D., Hines, J.E., 1997. Estimating temporary emigration using capture-recapture data with Pollock's robust design. *Ecology* 78, 563–578.
- Kendall, W.L., White, G.C., 2009. A cautionary note on substituting spatial subunits for repeated temporal sampling in studies of site occupancy. *J. Appl. Ecol.* 46, 1182–1188.
- Kéry, M., 2002. Inferring the absence of a species—a case study of snakes. *J. Wildl. Manage.* 66, 330–338.
- Kéry, M., 2004. Extinction rate estimates for plant populations in revisit studies: importance of detectability. *Conserv. Biol.* 18, 570–574.
- Kéry, M., 2008. Estimating abundance from bird counts: binomial mixture models uncover complex covariate relationships. *Auk* 125, 336–345.
- Kéry, M., 2010. Introduction to WinBUGS for Ecologists. A Bayesian Approach to Regression, ANOVA, Mixed Models and Related Analyses. Academic Press, Burlington, MA.
- Kéry, M., 2011a. Species richness and community dynamics—a conceptual framework. In: O'Connell, A.F., Nichols, J.D., Karanth, K.U. (Eds.), Camera Traps in Animal Ecology: Methods and Analyses. Springer, Tokyo, pp. 207–231.
- Kéry, M., 2011b. Towards the modeling of true species distributions. *J. Biogeogr.* 38, 617–618.
- Kéry, M., Dorazio, R.M., Soldaat, L., van Strien, A., Zuidewijk, A., Royle, J.A., 2009a. Trend estimation in populations with imperfect detection. *J. Appl. Ecol.* 46, 1163–1172.
- Kéry, M., Gardner, B., Monnerat, C., 2010a. Predicting species distributions from checklist data using site-occupancy models. *J. Biogeogr.* 37, 1851–1862.
- Kéry, M., Gardner, B., Stoeckle, T., Weber, D., Royle, J.A., 2011. Use of spatial capture-recapture modeling and DNA data to estimate densities of elusive animals. *Conserv. Biol.* 25, 356–364.
- Kéry, M., Gregg, K.B., 2003. Effects of life-state on detectability in a demographic study of the terrestrial orchid *Cleistes bifaria*. *J. Ecol.* 91, 265–273.
- Kéry, M., Gregg, K.B., 2004. Demographic analysis of dormancy and survival in the terrestrial orchid *Cypripedium reginae*. *J. Ecol.* 92, 686–695.
- Kéry, M., Gregg, K.B., Schaub, M., 2005a. Demographic estimation methods for plants with unobservable life-states. *Oikos* 108, 307–320.
- Kéry, M., Madsen, J., Lebreton, J.D., 2006. Survival of Svalbard pink-footed geese *Anser brachyrhynchus* in relation to winter climate, density and land-use. *J. Anim. Ecol.* 75, 1172–1181.
- Kéry, M., Royle, J.A., 2008. Hierarchical Bayes estimation of species richness and occupancy in spatially replicated surveys. *J. Appl. Ecol.* 45, 589–598.

- Kéry, M., Royle, J.A., 2009. Inference about species richness and community structure using species-specific occupancy models in the national Swiss breeding bird survey MHB. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*, Springer, New York, pp. 639–656.
- Kéry, M., Royle, J.A., 2010. Hierarchical modeling and estimation of abundance in metapopulation designs. *J. Anim. Ecol.* 79, 453–461.
- Kéry, M., Royle, J.A., Plattner, M., Dorazio, R.M., 2009b. Species richness and occupancy estimation in communities subject to temporary emigration. *Ecology* 90, 1279–1290.
- Kéry, M., Royle, J.A., Schmid, H., 2005b. Modeling avian abundance from replicated counts using binomial mixture models. *Ecol. Appl.* 15, 1450–1461.
- Kéry, M., Royle, J.A., Schmid, H., Schaub, M., Volet, B., Häfliger, G., Zbinden, N., 2010b. Site-occupancy distribution modeling to correct population-trend estimates derived from opportunistic observations. *Conserv. Biol.* 24, 1388–1397.
- Kéry, M., Schmidt, B.R., 2008. Imperfect detection and its consequences for monitoring for conservation. *Comm. Ecol.* 9, 207–216.
- King, R., Brooks, S.P., 2002. Bayesian model discrimination for multiple strata capture-recapture data. *Biometrika* 89, 785–806.
- King, R., Brooks, S.P., 2004. Bayesian analyses of the Hector's dolphin data. *Anim. Biodiv. Cons.* 27.1, 343–354.
- King, R., Morgan, B.J.T., Gimenez, O., Brooks, S.P., 2010. *Bayesian Analysis for Population Ecology*. Chapman & Hall, Boca Raton, FL.
- Knape, J., 2008. Estimability of density dependence in models of time series data. *Ecology* 89, 2994–3000.
- Knape, J., Jonzén, N., Sköld, M., 2011. On observation distributions for state space models of population survey data. *J. Anim. Ecol.* (in press).
- Krebs, C.J., 2001. *The Experimental Analysis of Distribution and Abundance*. Benjamin Cummings, San Francisco, CA.
- Krebs, C.J., Davies, N.B., 1993. *An Introduction to Behavioural Ecology*. Blackwell Scientific, Oxford.
- Lande, R., 2002. Incorporating stochasticity in population viability analysis. In: Beissinger, S.R. (Eds.), *Population Viability Analysis*. The University of Chicago Press, Chicago, IL, pp. 18–40.
- Lande, R., Engen, S., Saether, B.E., 2003. *Stochastic Population Dynamics in Ecology and Conservation*. Oxford University Press, Oxford.
- Lebreton, J.D., 2009. Assessing density-dependence: where are we left? In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 19–42.
- Lebreton, J.D., Almeras, T., Pradel, R., 1999. Competing events, mixtures of information and multistratum recapture models. *Bird Study* 46, 39–46.
- Lebreton, J.D., Burnham, K.P., Clobert, J., Anderson, D.R., 1992. Modeling survival and testing biological hypothesis using marked animals: a unified approach with case studies. *Ecol. Monogr.* 62, 67–118.
- Lebreton, J.D., Morgan, B.J.T., Pradel, R., Freeman, S.N., 1995. A simultaneous survival rate analysis of dead recovery and live recapture data. *Biometrics* 51, 1418–1428.
- Lebreton, J.D., Nichols, J.D., Barker, R.J., Pradel, R., Spendelow, J.A., 2009. Modeling individual animal histories with multistate capture-recapture models. *Adv. Ecol. Res.* 41, 87–173.
- Lebreton, J.D., Pradel, R., 2002. Multistate recapture models: modelling incomplete individual histories. *J. Appl. Stat.* 29, 353–369.
- Le Cam, L., 1990. Maximum likelihood – an introduction. *ISI Rev.* 58, 153–171.
- Lee, Y., Nelder, J.A., 2000. Two ways of modeling overdispersion in non-normal data. *App. Stat.* 49, 591–598.

- Lee, Y., Nelder, J.A., 2006. Double hierarchical generalized linear models. *App. Stat.* 55, 139–185.
- Lele, S.R., Dennis, B., Lutscher, F., 2007. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. *Ecol. Lett.* 10, 551–563.
- Lindley, D.V., 1983. Theory and practice of Bayesian statistics. *Statistician* 32, 1–11.
- Lindley, D.V., 2006. Understanding Uncertainty. Wiley, Hoboken, NJ.
- Lindley, S.T., 2003. Estimation of population growth and extinction parameters from noisy data. *Ecol. Appl.* 13, 806–813.
- Link, W.A., 1999. Modeling pattern in collections of parameters. *J. Wildl. Manage.* 63, 1017–1027.
- Link, W.A., 2003. Nonidentifiability of population size from capture-recapture data with heterogeneous detection probabilities. *Biometrics* 59, 1123–1130.
- Link, W.A., Barker, R.J., 2005. Modeling association among demographic parameters in analysis of open population capture-recapture data. *Biometrics* 61, 46–54.
- Link, W.A., Barker, R.J., 2010. Bayesian Inference with Ecological Applications. Academic Press, London.
- Link, W.A., Nichols, J.D., 1994. On the importance of sampling variance to investigations of temporal variations in animal population size. *Oikos* 69, 539–544.
- Link, W.A., Royle, J.A., Hatfield, J.S., 2003. Demographic analysis from summaries of an age-structured population. *Biometrics* 59, 778–785.
- Link, W.A., Sauer, J.R., 1996. Extremes in ecology: avoiding the misleading effects of sampling variation in summary analyses. *Ecology* 77, 1633–1640.
- Link, W.A., Sauer, J.R., 1998. Estimating population change from count data: application to the North American breeding bird survey. *Ecol. Appl.* 8, 258–268.
- 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., Yoshizaki, J., Bailey, L.L., Pollock, K.H., 2010. Uncovering a latent multinomial: analysis of mark-recapture data with misidentification. *Biometrics* 66, 178–185.
- Little, R.J.A., 2006. Calibrated Bayes: A bayes/frequentist roadmap. *Am. Stat.* 60, 213–223.
- Liu, J.S., Wu, Y.N., 1999. Parameter expansion for data augmentation. *J. Am. Stat. Assoc.* 94, 1264–1274.
- Lukacs, P.M., Burnham, K.P., 2005. Estimating population size from DNA-based closed capture–recapture data incorporating genotyping error. *J. Wildl. Manage.* 69, 396–403.
- Lunn, D.J., Spiegelhalter, D., Thomas, A., Best, N., 2009. The BUGS project: evaluation, critique and future directions. *Stat. Med.* 28, 3049–3067.
- Lunn, D.J., Thomas, A., Best, N., Spiegelhalter, D., 2000. WinBUGS—a Bayesian modelling framework: concepts, structure, and extensibility. *Stat. Comput.* 10, 325–337.
- MacKenzie, D.I., 2005. What are the issues with presence-absence data for wildlife managers? *J. Wildl. Manage.* 69, 849–860.
- MacKenzie, D.I., 2006. Modeling the probability of resource use: the effect of, and dealing with, detecting a species imperfectly. *J. Wildl. Manage.* 70, 367–374.
- MacKenzie, D.I., Nichols, J.D., Hines, J.E., Knutson, M.G., Franklin, A.B., 2003. Estimating site occupancy, colonization, and local extinction when a species is detected imperfectly. *Ecology* 84, 2200–2207.
- MacKenzie, D.I., Nichols, J.D., Lachman, G.B., Droege, S., Royle, J.A., Langtimm, C.A., 2002. Estimating site occupancy rates when detection probabilities are less than one. *Ecology* 83, 2248–2255.
- MacKenzie, D.I., Nichols, J.D., Royle, J.A., Pollock, K.H., Hines, J.E., Bailey, L.L., 2006. Occupancy Estimation and Modeling: Inferring Patterns and Dynamics of Species Occurrence. Elsevier, San Diego, CA.

- MacKenzie, D.I., Nichols, J.D., Seamans, M.E., Gutierrez, R.J., 2009. Modeling species occurrence dynamics with multiple states and imperfect detection. *Ecology* 90, 823–835.
- Marra, P.P., Griffing, S., Caffrey, C., Kilpatrick, A.M., McLean, R., Brand, C., et al., 2004. West nile virus and wildlife. *BioScience* 54, 393–402.
- Marshall, M.R., Diefenbach, D.R., Wood, L.A., Cooper, R.J., 2004. Annual survival estimation of migratory songbirds confounded by incomplete breeding site-fidelity: study designs that may help. *Anim. Biodiv. Cons.* 27.1, 59–72.
- Martin, J.E., Royle, J.A., Gardner, B., MacKenzie, D.I., Edwards, H.H., Kéry, M., 2011. Accounting for non-independent detection when estimating abundance of organisms with a Bayesian approach. *Meth. Ecol. Evol.* (in press).
- Martin, T.E., Clobert, J., Anderson, D.R., 1995. Return rates in studies of life history evolution: are biases large? *J. Appl. Stat.* 22, 863–875.
- Maunder, M.N., 2004. Population viability analysis based on combining Bayesian, integrated, and hierarchical analyses. *Acta Oecol.* 26, 85–94.
- McCarthy, M.A., 2007. *Bayesian Methods for Ecology*. Cambridge University Press, Cambridge.
- McCarthy, M.A., Masters, P., 2005. Profiting from prior information in Bayesian analyses of ecological data. *J. Appl. Ecol.* 42, 1012–1019.
- McClintock, B.T., Nichols, J.D., Bailey, L.L., MacKenzie, D.I., Kendall, W.L., Franklin, A.B., 2010. Seeking a second opinion: uncertainty in disease ecology. *Ecol. Lett.* 13, 659–674.
- McCullagh, P., Nelder, J.A., 1989. *Generalized Linear Models*. Chapman & Hall, London.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys.* 21, 1087–1092.
- Millar, R.B., 2009. Comparison of hierarchical Bayesian models for overdispersed count data using DIC and Bayes' factors. *Biometrics* 65, 962–969.
- Miller, D.A., Nichols, J.D., McClintock, B.T., Grant, E.H.C., Bailey, L.L., Weir, L., 2011. Improving occupancy estimation when two types of observational errors occur: non-detection and species misidentification. *Ecology* 92, 1422–1428.
- Moilanen, A., 2002. Implications of empirical data quality to metapopulation model parameter estimation and application. *Oikos* 96, 516–530.
- Monneret, R.-J., 2006. *Le faucon pèlerin*. Delachaux and Niestlé, Paris.
- Moritz, C., Patton, J.L., Conroy, C.J., Parra, J.L., White, G.C., Beissinger, S.R., 2008. Impact of a century of climate change on small mammal communities in Yosemite National Park, USA. *Science*, 322, 261–264.
- Newman, K.B., Buckland, S.T., Lindley, S.T., Thomas, L., Fernandez, C., 2006. Hidden process models for animal population dynamics. *Ecol. Appl.* 16, 74–86.
- Newton, I., 1998. *Population Limitation in Birds*. Academic Press, London.
- Nichols, J.D., Blohm, R.J., Reynolds, R.E., Trost, R.E., Hines, J.E., Bladen, J.P., 1991. Band reporting rates for Mallards with reward bands of different Dollar values. *J. Wildl. Manage.* 55, 119–126.
- Nichols, J.D., Boulinier, T., Hines, J.E., Pollock, K.H., Sauer, J.R., 1998a. Estimating rates of local species extinction, colonization, and turnover in animal communities. *Ecol. Appl.* 8, 1213–1225.
- Nichols, J.D., Boulinier, T., Hines, J.E., Pollock, K.H., Sauer, J.R., 1998b. Inference methods for spatial variation in species richness and community composition when not all species are detected. *Conserv. Biol.* 12, 1390–1398.
- Nichols, J.D., Hines, J.E., Lebreton, J.D., Pradel, R., 2000. Estimation of contributions to population growth: a reverse-time capture-recapture approach. *Ecology* 81, 3362–3376.
- Nichols, J.D., Hines, J.E., MacKenzie, D.I., Seamans, M.E., Gutierrez, R.J., 2007. Occupancy estimation and modeling with multiple states and state uncertainty. *Ecology* 88, 1395–1400.

- Nichols, J.D., Hines, J.E., Pollock, K.H., 1984. Effects of permanent trap response in capture probability on Jolly-Seber capture-recapture model estimates. *J. Wildl. Manage.* 48, 289–293.
- Nichols, J.D., Kendall, W.L., Hines, J.E., Spendelow, J.A., 2004. Estimation of sex-specific survival from capture-recapture data when sex is not always known. *Ecology* 85, 3192–3201.
- Nichols, J.D., Pollock, K.H., 1983. Estimation methodology in contemporary small mammal capture-recapture studies. *J. Mamm.* 64, 253–260.
- Nichols, J.D., Pollock, K.H., 1990. Estimation of recruitment from immigration versus in situ reproduction using Pollock's Robust design. *Ecology* 71, 21–26.
- Nichols, J.D., Thomas, L., Conn, P.B., 2009. Inferences about landbird abundance from count data: recent advances and future directions. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 201–235.
- Norris, K., 2004. Managing threatened species: the ecological toolbox, evolutionary theory and declining-population paradigm. *J. Appl. Ecol.* 41, 413–426.
- Ntzoufras, I., 2009. *Bayesian Modeling Using WinBUGS*. Wiley, Hoboken, NJ.
- O'Hara, R.B., Sillanpää, M.J., 2009. A review of Bayesian variable selection methods: what, how and which. *Bayesian Anal.* 4, 85–118.
- Orme, C.D.L., Davies, R.G., Burgess, M., Eigenbrod, F., Pickup, N., Olson, V.A., et al., 2005. Global hotspots of species richness are not congruent with endemism or threat. *Nature* 436, 1019.
- Otis, D.L., Burnham, K.P., White, G.C., Anderson, D.R., 1978. Statistical inference from capture data on closed animal populations. *Wildl. Monogr.* 62, 1–135.
- Pagel, J., Schurr, F.M., 2011. Forecasting species ranges by statistical estimation of ecological niches and spatial population dynamics. *Global Ecol. Biogeogr.* (in press).
- Pearman, P.B., Weber, D., 2007. Common species determine richness patterns in biodiversity indicator taxa. *Biol. Cons.* 138, 109–119.
- Pellet, J., Schmidt, B.R., 2005. Monitoring distributions using call surveys: estimating site occupancy, detection probabilities and inferring absence. *Biol. Cons.* 123, 27–35.
- Péron, G., Crochet, P.-A., Doherty, P.F., Lebreton, J.D., 2010. Studying dispersal at the landscape scale: efficient combination of population surveys and capture-recapture data. *Ecology* 91, 3365–3375.
- Phillips, S.J., Dudik, M., 2008. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. *Ecography* 31, 161–175.
- Pigliucci, M., 2002. *Denying Evolution: Creationism, Scientism, and the Nature of Science*. Sinauer Associates, Sunderland, MA.
- Pledger, S., 2000. Unified maximum likelihood estimates for closed capture-recapture models using mixtures. *Biometrics* 56, 434–442.
- Pledger, S., Efford, M., 1998. Correction of bias due to heterogeneous capture probability in capture-recapture studies of open populations. *Biometrics* 54, 888–898.
- Pledger, S., Efford, M., Pollock, K., Collazo, J., Lyons, J., 2009. Stopover duration analysis with departure probability dependent on unknown time since arrival. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 349–363.
- Pollock, K.H., 1982. A capture-recapture design robust to unequal probability of capture. *J. Wildl. Manage.* 46, 752–757.
- Pollock, K.H., Nichols, J.D., Brownie, C., Hines, J.E., 1990. Statistical inference for capture-recapture experiments. *Wildl. Monogr.* 107, 1–97.
- Post van der Burg, M., Bly, B., Vercauteren, T., Tyre, A.J., 2011. Making better use of monitoring data from low density species using a spatially explicit modeling approach. *J. Appl. Ecol.* 48, 47–55.
- Powell, L.A., 2007. Approximating variance of demographic parameters using the delta method: a reference for avian biologists. *Condor* 109, 949–954.

- Pradel, R., 1993. Flexibility in survival analysis from recapture data: handling trap-dependence. In: Lebreton, J.D. (Ed.), *Marked Individuals in the Study of Bird Population*. Birkhäuser-Verlag, Basel, pp. 29–37.
- Pradel, R., 1996. Utilization of capture-mark-recapture for the study of recruitment and population growth rate. *Biometrics* 52, 703–709.
- Pradel, R., 2005. Multievent: an extension of multistate capture-recapture models to uncertain states. *Biometrics* 61, 442–447.
- Pradel, R., Hines, J.E., Lebreton, J.D., Nichols, J.D., 1997. Capture-recapture survival models taking account of transients. *Biometrics* 53, 60–72.
- Pradel, R., Lebreton, J.D., 1999. Comparison of different approaches to the study of local recruitment of breeders. *Bird Study* 46, 74–81.
- Pradel, R., Wintrebert, C.M.A., Gimenez, O., 2003. A proposal for a goodness-of-fit test to the Arnason-Schwarz multistate capture-recapture model. *Biometrics* 59, 43–53.
- Purvis, A., Hector, A., 2000. Getting the measure of biodiversity. *Nature* 405, 212–219.
- R Development Core Team, 2004. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna.
- Reid, J.M., Bignal, E.M., Bignal, S., McCracken, D.I., Monaghan, P., 2003. Environmental variability, life-history covariation and cohort effects in the red-billed chough *Pyrrhocorax pyrrhocorax*. *J. Anim. Ecol.* 72, 36–46.
- Reif, J., Storch, D., Simova, I., 2008. The effect of scale-dependent habitat gradients on the structure of bird assemblages in the Czech Republic. *Acta Ornithol.* 43, 197–206.
- Rexstad, E., Burnham, K.P., 1991. User's Guide for Interactive Program CAPTURE. Colorado Cooperative Fish & Wildlife Research Unit, Colorado State University, Fort Collins, CO.
- Risk, B.B., De Valpine, P., Beissinger, S.R., 2011. A robust-design formulation of the incidence function model of metapopulation dynamics applied to two rail species. *Ecology* 92, 462–474.
- Robinson, R.A., Green, R.E., Baillie, S.R., Peach, W.J., Thomson, D.L., 2004. Demographic mechanisms of the population decline of the song thrush *Turdus philomelos* in Britain. *J. Anim. Ecol.* 73, 670–682.
- Rota, C.T., Fletcher Jr., R.J., Dorazio, R.M., Betts, M.G., 2009. Occupancy estimation and the closure assumption. *J. Appl. Ecol.* 46, 1173–1181.
- Roth, T., Amrhein, V., 2009. Estimating individual survival using territory occupancy data on unmarked animals. *J. Appl. Ecol.* 47, 386–392.
- Royle, J.A., 2004a. Generalized estimators of avian abundance from count survey data. *Anim. Biodiv. Cons.* 27.1, 375–386.
- Royle, J.A., 2004b. Modeling abundance index data from anuran calling surveys. *Conserv. Biol.* 18, 1378–1385.
- Royle, J.A., 2004c. N-mixture models for estimating population size from spatially replicated counts. *Biometrics* 60, 108–115.
- Royle, J.A., 2006. Site occupancy model with heterogeneous detection probabilities. *Biometrics* 62, 97–102.
- Royle, J.A., 2008. Modeling individual effects in the Cormack-Jolly-Seber model: a state-space formulation. *Biometrics* 64, 364–370.
- Royle, J.A., 2009. Analysis of capture-recapture models with individual covariates using data augmentation. *Biometrics* 65, 267–274.
- Royle, J.A., Dorazio, R.M., 2006. Hierarchical models of animal abundance and occurrence. *JABES* 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., Dorazio, R.M., 2011. Parameter-expanded data augmentation for Bayesian analysis of capture-recapture models. *J. Ornithol.* (in press).

- Royle, J.A., Dorazio, R.M., Link, W.A., 2007a. Analysis of multinomial models with unknown index using data augmentation. *J. Comput. Stat.* 16, 67–85.
- Royle, J.A., Dubovsky, J.A., 2001. Modeling spatial variation in waterfowl band-recovery data. *J. Wildl. Manage.* 65, 726–737.
- Royle, J.A., Gardner, B., 2011. Hierarchical spatial capture-recapture models for estimating density from trap-arrays. In: O'Connell, A.F., Nichols, J.D., Karanth, K.U. (Eds.), *Camera Traps in Animal Ecology—Methods and Analyses*. Springer, New York, pp. 163–190.
- Royle, J.A., Karanth, K.U., Gopalaswamy, A.M., Kumar, N.S., 2009a. Bayesian inference in camera trapping studies for a class of spatial capture-recapture models. *Ecology* 90, 3233–3244.
- Royle, J.A., Kéry, M., 2007. A Bayesian state-space formulation of dynamics occupancy models. *Ecology* 88, 1813–1823.
- Royle, J.A., Kéry, M., Gauthier, R., Schmid, H., 2007b. Hierarchical spatial models of abundance and occurrence from imperfect survey data. *Ecol. Monogr.* 77, 465–481.
- Royle, J.A., Kéry, M., Guélat, J., 2011. Spatial capture-recapture models for search-encounter data. *Meth. Ecol. Evol.* (in press).
- Royle, J.A., Link, W.A., 2005. A general class of multinomial mixture models for anuran calling survey data. *Ecology* 86, 2505–2512.
- Royle, J.A., Link, W.A., 2006. Generalized site occupancy models allowing for false positive and false negative errors. *Ecology* 87, 835–841.
- Royle, J.A., Nichols, J.D., 2003. Estimating abundance from repeated presence-absence data or point counts. *Ecology* 84, 777–790.
- Royle, J.A., Nichols, J.D., Karanth, K.U., Gopalaswamy, A.M., 2009b. A hierarchical model for estimating density in camera-trap studies. *J. Appl. Ecol.* 46, 118–127.
- 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., Young, K.G., 2008. A hierarchical model for spatial capture-recapture data. *Ecology* 89, 2281–2289.
- Rue, H., Martino, S., Chopin, N., 2009. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations (with discussion). *J. R. Stat. Soc. B* 71, 319–392.
- Ruiz-Gutierrez, V., Zipkin, E.F., 2011. Detection biases yield misleading patterns of species persistence and colonization in fragmented landscapes. *Ecosphere* 2, Article 61.
- Russell, R.E., Royle, J.A., Saab, V.A., Lehmkuhl, J.F., Block, W.M., Sauer, J.A., 2009. Modeling the effects of environmental disturbance on wildlife communities: avian responses to prescribed fire. *Ecol. Appl.* 19, 1253–1263.
- Sæther, B.E., Bakke, O., 2000. Avian life history variation and contribution of demographic traits to the population growth rate. *Ecology* 81, 642–653.
- Saracco, J.F., Royle, J.A., DeSante, D.F., Gardner, B., 2010. Modeling spatial variation in avian survival and residency probabilities. *Ecology* 91, 1885–1891.
- Sauer, J.R., Link, W.A., 2002. Hierarchical modeling of population stability and species group attributes from survey data. *Ecology* 86, 1743–1751.
- Schaub, M., 2009. Estimation of cause-specific mortality rates from ring recovery data: a Bayesian evaluation. In: Thomson, D.L., Cooch, E.G., Conroy, M.J. (Eds.), *Modeling Demographic Processes in Marked Populations*. Springer, New York, pp. 1081–1097.
- Schaub, M., Abadi, F., 2011. Integrated population models: A novel analysis framework for deeper insights into population dynamics. *J. Ornithol.* (in press).
- Schaub, M., Aebsischer, A., Gimenez, O., Berger, S., Arlettaz, R., 2010. Massive immigration balances high human induced mortality in a stable eagle owl population. *Biol. Cons.* 143, 1911–1918.
- Schaub, M., Gimenez, O., Schmidt, B.R., Pradel, R., 2004a. Estimating survival and temporary emigration in the multistate capture-recapture framework. *Ecology* 85, 2107–2113.

- Schaub, M., Gimenez, O., Sierro, A., Arlettaz, R., 2007. Use of integrated modeling to enhance estimates of population dynamics obtained from limited data. *Conserv. Biol.* 21, 945–955.
- Schaub, M., Liechti, F., Jenni, L., 2004b. Departure of migrating European robins, *Erithacus rubecula*, from a stopover site in relation to wind and rain. *Anim. Behav.* 67, 229–237.
- Schaub, M., Pradel, R., 2004. Assessing the relative importance of different sources of mortality from recoveries of marked animals. *Ecology* 85, 930–938.
- Schaub, M., Reichlin, T.S., Abadi, F., Kéry, M., Jenni, L., Arlettaz, R., 2011. The demographic drivers of local population dynamics in two rare migratory birds. *Oecologia* (in press).
- Schaub, M., Ullrich, B., Knötzsch, G., Albrecht, P., Meisser, C., 2006. Local population dynamics and the impact of scale and isolation: a study on different little owl populations. *Oikos* 115, 389–400.
- Schaub, M., Zink, R., Beissmann, H., Sarrazin, F., Arlettaz, R., 2009. When to end releases in reintroduction programmes: demographic rates and population viability analysis of bearded vultures in the Alps. *J. Appl. Ecol.* 46, 92–100.
- Schlossberg, S., King, D.I., Chandler, R.B., Mazzei, D.A., 2010. Regional synthesis of habitat relationships in shrubland birds. *J. Wildl. Manage.* 74, 1513–1522.
- Schmid, H., Luder, R., Naef-Daenzer, B., Graf, R., Zbinden, N., 1998. Schweizer Brutvogelatlas. Schweizerische Vogelwarte, Sempach, Switzerland.
- Schmid, H., Zbinden, N., Keller, V., 2004. Überwachung der Bestandsentwicklung häufiger Brutvögel in der Schweiz. Schweizerische Vogelwarte, Sempach, Switzerland.
- Schmidt, B.R., 2005. Monitoring the distribution of pond-breeding amphibians when species are detected imperfectly. *Aquat. Conserv.: Mar. Freshw. Ecosyst.* 15, 681–692.
- Schmidt, B.R., Feldmann, R., Schaub, M., 2005. Demographic processes underlying population growth and decline in *Salamandra salamandra*. *Conserv. Biol.* 19, 1149–1156.
- Schofield, M.R., Barker, R.J., 2008. A unified capture-recapture framework. *JABES* 13, 458–477.
- Schofield, M.R., Barker, R.J., MacKenzie, D.I., 2009. Flexible hierarchical mark-recapture modeling for open populations using WinBUGS. *Environ. Ecol. Stat.* 16, 369–387.
- Schorcht, W., Bontadina, F., Schaub, M., 2009. Variation of adult survival drives population dynamics in a migrating forest bat. *J. Anim. Ecol.* 78, 1182–1190.
- Schwarz, C.J., 2002. Real and quasi-experiments in capture-recapture studies. *J. Appl. Stat.* 29, 459–473.
- Schwarz, C.J., Arnason, A.N., 1996. A general methodology for the analysis of capture-recapture experiments in open populations. *Biometrics* 52, 860–873.
- Schwarz, C.J., Schweigert, J.F., Arnason, A.N., 1993. Estimating migration rates using tag recovery data. *Biometrics* 49, 177–193.
- Scott, J.M., Heglund, P.J., Haufler, J.B., Morrsion, M., Raphael, M.G., Wall, W.B., et al., 2002. Predicting Species Occurrence: Issues of Accuracy and Scale. Island Press, Covelo, CA.
- Seber, G.A.F., 1965. A note on the multiple recapture census. *Biometrika* 52, 249–259.
- Seber, G.A.F., 1982. The Estimation of Animal Abundance and Related Parameters. Charles Griffin & Company Ltd, London.
- Seber, G.A.F., Wild, C.J., 2003. Nonlinear Regression. Wiley, Hoboken, NJ.
- Servanthy, S., Choquet, R., Baubet, E., Brandt, S., Gaillard, J.M., Schaub, M., et al., 2010. Assessing whether mortality is additive using marked animals: a Bayesian state-space modeling approach. *Ecology* 91, 1916–1923.
- Service, P.M., 2000. Heterogeneity in individual mortality risk and its importance for evolutionary studies of senescence. *Am. Nat.* 156, 1–13.
- Shefferson, R.P., Sandercock, B.K., Proper, J., Beissinger, S.R., 2001. Estimating dormancy and survival of a rare herbaceous perennial using mark-recapture models. *Ecology* 82, 145–156.
- Sibly, R.M., Calow, P., 1986. Physiological Ecology of Animals: An Evolutionary Approach. Blackwell Scientific, Oxford.

- Sibly, R.M., Hone, J., 2002. Population growth rate and its determinants: an overview. *Phil. Trans. R. Soc. B* 357, 1153–1170.
- Smith, A.F.M., Gelfand, A., 1993. Bayesian statistics without tears. *Am. Stat.* 46, 84–88.
- Smout, S., King, R., Pomeroy, P.P., 2011. Integrating heterogeneity of detection and mark loss to estimate survival and transience in UK grey seal colonies. *J. Appl. Ecol.* 48, 364–372.
- Spiegelhalter, D.J., 1998. Bayesian graphical modelling: a case-study in monitoring health outcomes. *App. Stat.* 47, 115–133.
- Spiegelhalter, D.J., Best, N.G., Carlin, B.P., van der Linde, A., 2002. Bayesian measure of model complexity and fit. *J. R. Stat. Soc. B* 64, 583–639.
- Stearns, S.C., 1992. *The Evolution of Life Histories*. Oxford University Press, Oxford.
- Sturtz, S., Ligges, U., Gelman, A., 2005. R2WinBUGS: a package for running WinBUGS from R. *J. Stat. Softw.* 12, 1–16.
- Sutherland, B.J., Dolman, P.M., 1994. Combining behaviour and population dynamics with applications for predicting consequences of habitat loss. *Proc. R. Soc. Lond. B* 255, 133–138.
- Talley, T.S., Fleishman, E., Holyoak, M., Murphy, D.D., Ballard, A., 2007. Rethinking a rare-species conservation strategy in an urban landscape: the case of the valley elderberry longhorn beetle. *Biol. Cons.* 135, 21–32.
- Tanner, M.A., Wong, W.H., 1987. The calculation of posterior distributions by data augmentation. *J. Am. Stat. Assoc.* 82, 528–540.
- Tavecchia, G., Besbeas, P., Coulson, T., Morgan, B.J.T., Clutton-Brock, T.H., 2009. Estimating population size and hidden demographic parameters with state-space modeling. *Am. Nat.* 173, 722–733.
- Tavecchia, G., Pradel, R., Gossmann, F., Bastat, C., Ferrand, Y., Lebreton, J.D., 2002. Temporal variation in annual survival probability of the Eurasian woodcock *Scolopax rusticola* wintering in France. *Wildl. Biol.* 8, 21–30.
- Thomas, C.D., Lennon, J.J., 1999. Birds extend their ranges northwards. *Nature* 399, 213.
- Thomas, L., Buckland, S.T., Newman, K.B., Harwood, J., 2005. A unified framework for modelling wildlife population dynamics. *Aust. N. Z. J. Stat.* 47, 19–34.
- Thompson, D.K., 2007. Use of site-occupancy models to estimate prevalence of *Myxobolus cerebralis* infection in trout. *J. Anim. Health* 19, 8–13.
- Thompson, S.K., 2002. *Sampling*. Wiley, New York.
- Tingley, M.W., Beissinger, S.R., 2009. Detecting range shifts from historical species occurrences: new perspectives on old data. *Trend. Ecol. Evol.* 24, 625–633.
- Tingley, M.W., Monahan, W.B., Beissinger, S.R., Moritz, C., 2009. Birds track their Grinnellian niche through a century of climate change. *Proc. Nat. Acad. Sci. USA* 106, 19637–19643.
- Tyre, A.J., Tenhumberg, B., Field, S.A., Niejalke, D., Parris, K., Possingham, H.P., 2003. Improving precision and reducing bias in biological surveys: estimating false-negative error rates. *Ecol. Appl.* 13, 1790–1801.
- van de Pol, M., Verhulst, S., 2006. Age-dependent traits: a new statistical model to separate within- and between-individual effects. *Am. Nat.* 167, 766–773.
- van Strien, A.J., van Swaay, C.A.M., Kéry, M., 2011. Metapopulation dynamics in the butterfly *Hipparchia semele* changed decades before decline in the Netherlands. *Ecol. Appl.* (in press).
- Ver Hoef, J.M., Jansen, J.K., 2007. Space-time zero-inflated count models of harbour seals. *Environmetrics* 18, 697–712.
- Weber, D., Hintermann, U., Zanger, A., 2004. Scale and trends in species richness: considerations for monitoring biological diversity for political purposes. *Glob. Ecol. Biogeogr.* 13, 97–104.
- Webster, R.A., Pollock, K.H., Simons, T.R., 2008. Bayesian spatial modeling of data from avian point surveys. *JABES* 13, 121–139.
- Welham, S., Cullins, B., Gogel, B., Gilmour, A., Thompson, R., 2004. Prediction in linear mixed models. *Aust. N. Z. J. Stat.* 46, 325–347.

- 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.
- White, G.C., Garrott, R.A., 1990. Analysis of Wildlife Radio-Tracking Data. Academic Press, London.
- Wikle, C.K., 2003. Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology* 84, 1382–1394.
- Williams, B.K., Nichols, J.D., Conroy, M.J., 2002. Analysis and Management of Animal Populations. Academic Press, San Diego, CA.
- Wright, J.A., Barker, R.J., Schofield, M.R., Frantz, A.C., Byrom, A.E., Gleeson, D.M., 2009. Incorporating genotype uncertainty into mark-recapture-type models for estimating abundance using DNA samples. *Biometrics* 65, 833–840.
- Yamaura, Y., Royle, J.A., Kubio, K., Tada, T., Ikeno, S., Makino, S., 2011. Modelling community dynamics based on species-level abundance models from detection/nondetection data. *J. Appl. Ecol.* 48, 67–75.
- Yoshizaki, J., Pollock, K.H., Brownie, C., Webster, R.A., 2009. Modeling misidentification errors in capture-recapture studies using photographic identification of evolving marks. *Ecology* 90, 3–9.
- Zheng, C., Ovaskainen, O., Saastamoinen, M., Hanski, I., 2007. Age-dependent survival analyzed with Bayesian models of mark-recapture data. *Ecology* 88, 1970–1976.
- Zipkin, E.F., DeWan, A., Royle, J.A., 2009. Impacts of forest fragmentation on species richness: a hierarchical approach to community modelling. *J. Appl. Ecol.* 46, 815–822.
- Zipkin, E.F., Royle, J.A., Dawson, D.K., Bates, S., 2010. Multi-species occurrence models to evaluate the effects of conservation and management actions. *Biol. Cons.* 143, 479–484.

# Index

---

*Page numbers followed by f indicate figure, t indicate table*

## A

Abundance  
  apparent, 14t  
  covariate relationship with, 395, 395f  
  estimation, 15, 384  
    analysis of real data, 396–409  
    covariates in, 390–396  
    ecological process in, 392f  
    generation and analysis of simulated data, 388–396  
    GLMM and, 383  
    introduction to, 383–388  
    N-mixture model with  
    overdispersion in, 404–409  
    observation process in, 392f  
    open-population N-mixture model in, 396–409  
    posterior distributions in, 394f, 397, 401f  
    program R in, 396  

-values in, 397  
    silver-washed fritillary example of, 396–409, 397f, 408f  
    simple Poisson model for, 398–401  
    simplest case of, 388–390  
    summary of, 409–410  
    WinBUGS and, 388–410  
    ZIP N-mixture model in, 401–404  
false negatives and, 384  
importance of, 20  
as key population descriptor, 409–410  
latent, 384  
modeled, 14t  
overdispersion in, 404–409  
parameters, 387  
Poisson distribution for, 385  
spatial variation in, 384  
in species–abundance distribution, 5  
as state variable, 410, 414  
study of, 1–6  
true, 14t

Additive fixed effects, 154  
Age effects  
  models with, 208–212  
  time effects combined with, 212  
Age-dependent models, multinomial likelihood and, 227–231, 252–255  
Age-specific probability, of first breeding model  
  analysis of, 290–294  
  description of, 288–289  
  generation of simulated data, 289–290  
  survival and movement estimation with, 288–294  
Akaike's information criterion (AIC), 42, 469, 470  
Algebra  
  BUGS language and, 50–51, 84  
  modeling in, 25, 267  
  multistate models and, 267  
ANCOVA linear model  
  GLM, 50, 52  
  GLMM, 74–77, 106  
  R code for, 50, 52  
  random-effects, 75–76  
Andrewartha, H. G., 1  
ANOVA, 100–106  
Apparent abundance, 14t  
Apparent distribution, 14t, 415  
Asp viper, 17–18, 18–19f, 74–77, 76f  
Atlas, distribution, 4

## B

Band-recovery models. *See* Mark-recovery models  
Batched effects, pooling of, 81  
Bayes rule  
  exaggerated difficulty of, 36  
  information combined with, 35  
  learning formalized in, 34–35  
  paraphrased, 34

- Bayes rule (*Cont.*)  
 priors and, 35–37, 44  
 quantities in, 33
- Bayesian computation, in Bayesian statistical modeling, 38
- Bayesian statistical modeling  
 Bayesian computation in, 38  
 challenges of, 41  
 hierarchical, 43–44  
 inference in, 44  
 introduction to, 23–24, 28–38  
 overview of, 464  
 parameter nonidentifiability and, 217  
 posterior sampling and, 41–43  
*p*-values in, 222–223, 226f  
 role of, 24–27  
 summary of, 44–45  
 WinBUGS and, 38–40
- Begon, M., 2
- Behavioral effects  
 defined, 136  
 in generation and analysis of simulated data, 148–150  
 importance of, 138
- Bernoulli distribution  
 defined, 67  
 site-occupancy models and, 417
- Bernoulli trials, 176, 244
- Beta-binomial distribution, 12
- Bias  
 descriptor, 7  
 estimation error and, 366  
 simulations and, 16–20  
 systematic, 121–126
- Binomial distribution  
 beta-binomial, 12  
 capture–recapture models and, 385  
 negative, 12, 386  
 observation process and, 9–13, 20
- Binomial GLM  
 analysis of real data, 70–71  
 generation and analysis of simulated data, 68–70  
 for modeling-bounded counts or proportions, 67–71  
 Poisson GLM compared to, 48  
 summary of, 71–72  
 WinBUGS and, 68–71
- Binomial mixture model  
 abundance estimation and, 15, 384  
 analysis of real data, 396–409  
 covariates in, 390–396
- ecological process in, 392f  
 generation and analysis of simulated data, 388–396  
 GLMM and, 383  
 introduction to, 383–388  
 N-mixture model with overdispersion in, 404–409  
 observation process in, 392f  
 open-population N-mixture model in, 396–409  
 posterior distributions in, 394f, 397, 401f  
 program R in, 396  
*p*-values in, 397  
 silver-washed fritillary example of, 396–409, 397f, 408f  
 simple Poisson model for, 398–401  
 simplest case, 388–390  
 summary of, 409–410  
 WinBUGS and, 388–410  
 ZIP N-mixture model, 401–404
- assumptions, 387–388  
 benefits, 385  
 defined, 384  
 detection probability in, 387  
 for dynamic situations, 387  
 false positives and, 388  
 Poisson-binomial, 385–386  
 for static situations, 387
- Binomial response, 15
- Biodiversity, 4–5
- Birch, L. C., 1
- Black grouse, 329, 329f, 334f
- Blue bug. *See Rosalia alpina* beetle
- Brooks–Gelman–Rubin statistic, 39
- BRT. *See* Bugs run time
- bugs function, 87
- BUGS language  
 algebraic description in, 50–51, 84  
 as hierarchical model translation, 25  
 history of, 38  
 latent suitability indicators defined in, 402  
 likelihood defined in, 358  
 logit stabilized in, 430  
 value of, 25, 39, 41, 54–55
- Bugs run time (BRT), 93
- Burnin, 39

**C**

- Capture–recapture data  
 for estimating closed population size  
 analysis of real data, 157–162

- behavioral effects in, 148–150  
CJS model and, 135, 137–138  
combined effects in, 154–157  
generation and analysis of simulated data, 139–157  
individual effects in, 150–153  
introduction to, 134–139  
model  $M_{tbh}$  for, 157–162  
model  $M_{t+x}$  for, 162–168  
pen shell example of, 166–168, 167f  
summary of, 169–170  
time effects in, 145–147  
trap response example of, 136–137, 148–157  
in integrated population model example, 357–363  
joint analysis of  
  generation of simulated data, 296–297  
  model for, 295–300  
  survival and movement estimation from, 295–300  
likelihood of, 354  
mark-recovery data compared to, 295  
overdispersion and, 196  
for recruitment, survival, and  
  population size estimation  
  using JS model  
  analysis of real data, 341–345  
  assumptions of, 316  
  black grouse example of, 329, 329f, 334f  
  connections between parameters in, 339–341  
  constant survival and time-dependent entry models in, 328–335  
  data augmentation in, 319–328, 338  
  GLM and, 328, 345  
  grey-headed woodpecker example of, 335–339, 336f  
  identifiability in, 339–341  
  introduction to, 316–317  
  Leisler's bats example of, 341–345, 344f  
  models with individual capture heterogeneity, 335–339  
  parameterizations of, 317, 325–328  
  priors in, 326  
  summary of, 345–346  
  WinBUGS and, 321–339, 341–345  
sampling of, 472  
for survival and movement estimation  
  using multistate models  
  age-specific probability of first breeding model, 288–294  
  fire salamanders example of, 282–284, 283f  
  introduction to, 264–268  
  little ringed plover example of, 270–274, 271f, 289–290, 302–303  
  model for joint analysis of capture-recapture and mark-recovery data, 295–300  
  movement among three sites, 300–306  
  movement between two sites, 268–281  
  real-data example, 307–311  
  showy lady's slipper example, 307f, 307–311  
  state-space likelihood and, 274  
  summary of, 311–312  
  temporal effects and, 280  
  temporary emigration model, 281–287  
  WinBUGS and, 270–287, 289–294, 296–300, 302–312  
for survival estimation using CJS model  
  age effect models and, 208–212  
  age-dependent models and, 227–231  
  analysis of real data, 231–237  
  constant parameters and, 177–183  
  fixed group and random time effect models in, 204–208  
  fixed group and time effect models in, 199–204  
  fixed group effect models in, 192–194  
  fixed time effect models in, 184  
  GLM and, 199, 211, 237  
  individual group effects in, 195–199  
  introduction to, 172–175  
  latent state variable in, 181–183  
  Leisler's bat example, 231–237, 232f, 235–236f  
  little owls example of, 177–183, 178f, 185, 189, 196, 209, 227–231  
  models with individual variation and, 192–199  
  multinomial likelihood and, 220–231, 237  
  parameter identifiability in, 216–220  
  R code and, 178  
  random group effects in, 194–195

- Capture–recapture data (*Cont.*)
- random time effect models in, 184–188
  - recapture probability and, 212–216
  - red-backed shrikes example, 213, 214f
  - summary of, 237–238
  - temporal covariates in, 188–192
  - temporal variability, 204
  - time and group effect models in, 199–208
  - time-dependent models and, 222–227
  - time-variation models and, 183–192
  - trap response models and, 212–216
  - WinBUGS and, 177–208, 212–237
  - understanding of, 465
- Capture–recapture models
- advantages of, 169
  - binomial distribution and, 385
  - classical, 135–136
  - data augmentation and, 139–140, 152
  - detection probability and, 136–137, 137t
  - GLM and, 138
  - with individual covariates, 162–168
  - introduction to, 134
  - SECR, 135, 475
  - simulations with, 139–157
  - behavioral effects, 148–150
  - combined effects, 154–157
  - individual effects, 150–153
  - introduction to, 139–145
  - time effects, 145–147
  - WinBUGS and, 138–169
- Chance, 27
- Cheap data, 385, 409–410
- CI. *See* Confidence interval
- CJS model. *See* Cormack–Jolly–Seber model
- Classical capture–recapture models, 135–136
- Cleaner thinking, hierarchical models for, 467
- Closed population size, capture–recapture data for estimating
- analysis of real data, 157–162
  - behavioral effects in, 148–150
  - CJS model and, 135, 137–138
  - combined effects in, 154–157
  - generation and analysis of simulated data, 139–157
  - individual effects in, 150–153
  - introduction to, 134–139
  - model  $M_{t+1}$ , 157–162
  - model  $M_{t+X}$ , 162–168
- pen shell example of, 166–168, 167f
- summary of, 169–170
- time effects in, 145–147
- trap response example of, 136–137, 148–157
- Closure assumption
- relaxing of, 475–476
  - temporary emigration and, 416
- Coal tit, 95f, 95–110, 97f
- Coefficient of variation (CV), 129
- Common terns, 244–246, 245f, 252–255
- Community
- models, for metapopulation designs, 474–475
  - as population, 5
- Confidence interval (CI), 37
- Conservation, population analysis for, 477–478
- Constant detection probability, 447, 472
- Constant parameters, 177–183, 246–252
- Constant survival and time-dependent entry models
- comparison of estimates, 333–335
  - multistate, 332
  - recruitment, survival, and population size estimation using, 328–335
  - restricted occupancy, 331
  - superpopulation formulation and, 333
- Conventional Poisson GLMM, for count data analysis of real data, 88–90, 95–110
- fixed site and fixed year effects, 100–102
  - fixed site effects, 99–100
  - full model, 108–110
  - generation and analysis of simulated data, 84–88, 92–95
  - introduction to, 73–82
  - overdispersion in, 83
  - with random effects for variability among groups, 90–110
  - random site and random year effects, 103–105
  - random site effect, random year effect, and first-year fixed observer effect, 105–106
  - random site effect, random year effect, first-year fixed observer effect, and overall linear time trend, 106–108
  - random site effects, 102–103
  - uses of, 111
- Convergence, checking for, 62–63, 67, 70–71, 87, 89

- Cormack–Jolly–Seber (CJS) model  
assumptions, 173–174  
closed population size and, 135, 137–138  
emigration and, 371  
false negatives and, 173  
false positives and, 173  
in integrated population models, 371  
JS model compared to, 237, 345  
mark-recovery model compared to,  
    243–244, 248  
m-array and, 44, 220–231  
multinomial likelihood and, 173, 175,  
    220–231, 237  
multistate model compared to, 267, 278  
state-space models and, 173, 175–177, 181,  
    237, 465, 466t  
survival estimation using  
    age effect models and, 208–212  
    age-dependent models and, 227–231  
    analysis of real data, 231–237  
    constant parameters and, 177–183  
    fixed group and random time effect  
        models in, 204–208  
    fixed group and time effect models  
        in, 199–204  
    fixed group effect models in, 192–194  
    fixed time effect models in, 184  
    GLM and, 199, 211, 237  
    individual group effects in, 195–199  
    introduction to, 172–175  
    latent state variable in, 181–183  
Leisler's bat example, 231–237, 232f,  
    235–236f  
little owls example of, 177–183, 178f,  
    185, 189, 196, 209, 227–231  
models with individual variation  
and, 192–199  
multinomial likelihood and,  
    220–231, 237  
parameter identifiability in, 216–220  
R code and, 178  
random group effects in, 194–195  
random time effect models in,  
    184–188  
recapture probability and, 212–216  
red-backed shrikes example, 213,  
    214f  
summary of, 237–238  
temporal covariates in, 188–192  
temporal variability, 204  
time and group effect models in,  
    199–208  
time-dependent models and,  
    222–227  
time-variation models and, 183–192  
trap response models and, 212–216  
WinBUGS and, 177–208, 212–237
- Count data  
    in integrated population model, 357–363  
    likelihood of, 352–354  
Poisson GLMM for  
    analysis of real data, 88–90, 95–110  
    fixed site and fixed year effects,  
        100–102  
    fixed site effects, 99–100  
    full model, 108–110  
    generation and analysis of simulated  
        data, 84–88, 92–95  
    introduction to, 73–82  
    overdispersion in, 83  
    with random effects for variability  
        among groups, 90–110  
    random site and random year  
        effects, 103–105  
    random site effect, random year  
        effect, and first-year fixed observer  
        effect, 105–106  
    random site effect, random year  
        effect, first-year fixed observer effect,  
        and overall linear time trend,  
        106–108  
    random site effects, 102–103  
    uses of, 111  
replicated, 385, 409
- Covariates  
    in abundance estimation, 390–396  
    abundance's relationship with, 395, 395f  
    effects of, 79  
    flexible modeling of, 476  
    in individual covariate models  
        capture–recapture models, 162–168  
        for population size estimation,  
            166–168  
        for species richness estimation,  
            163–166  
    in N-mixture model, 390–396  
    single-season occupancy model with,  
        422–427  
    site, 390  
    temporal, 188–192
- Credible interval (CRI)  
    defined, 37  
    increase in, 369
- CV. *See* coefficient of variation

**D**

- DA. *See* Data augmentation
- DAG. *See* Directed acyclic graph
- Data augmentation (DA)
- capture–recapture models and, 139, 140, 152
  - defined, 140
  - generation and analysis of simulated data with, 139–157
    - behavioral effects, 148–150
    - combined effects, 154–157
    - individual effects, 150–153
    - introduction to, 139–145
    - time effects, 145–147
  - introduction to, 139
  - JS model and, 319–328, 338
    - as multistate model, 322–325
    - as restricted dynamic occupancy model, 320–322
    - superpopulation parameterization and, 325–328
  - in random-effects models, 152
  - in recruitment, survival, and population size estimation, 319–328, 338
- Data cloning, 36
- Data dredging, 469–470
- Dead-recovery matrix, 247
- Dead-recovery models. *See* Mark-recovery models
- Demographic parameters, temporal variability of, 373
- Demographic rates
- integrated population models estimating analysis of, 358–363
  - CJS model in, 371
  - fecundity estimated with, 363–366, 377–378
  - hoopoe example of, 371f, 371–379, 378f
  - introduction to, 348–350
  - ortolan bunting example of, 349, 350–351f, 357–363, 366–370
  - for population viability analysis, 366–370
  - without productivity data, 363–366
  - real data example of, 371–379
  - simple example of, 357–363
  - summary of, 379–380
  - WinBUGS and, 357–379
  - population size and, 350–352
  - Descriptors, biased, 7
- Detection. *See also* Detection/nondetection data
- linear predictor for, 386
  - N-mixture model with overdispersion with, 404–409
  - probability, 6, 11
    - in binomial mixture models, 387
    - capture–recapture models and, 137t, 136–137
    - constant, 447, 472
    - date related to, 433
    - model selection view of, 473
    - state-space models and, 124–125, 131
    - time of day related to, 433
    - within-capture-history dependence of, 148
  - of silver-washed fritillary, 408f
- Detection error. *See* False negatives
- Detection/nondetection data
- defined, 414
  - introduction to, 414–419
  - occurrence and species distribution estimation from
    - analysis of real data, 427–436, 445–450
    - dynamic models for, 436–450, 459
    - generation and analysis of simulated data in, 420–427, 439–445
    - goodness of fit and, 418
    - introduction to, 414–418
    - long-eared owl example of, 439–445, 440f, 454, 456
    - multistate occupancy models, 450–458
    - p*-values and, 419–420
    - Rosalia alpina* example of, 427–436, 428–429f, 432f, 434f
    - single-season occupancy analysis, 420–427
    - six-spot burnet example of, 445–450, 445f, 450f
    - summary of, 459–460
    - WinBUGS in, 419–436, 439–450
- Deterministic mechanisms, 6
- Deviance information criterion (DIC), 42, 469
- Directed acyclic graph (DAG), 356f, 467
- Dirichlet distribution, 300, 302, 308, 454
- Dispersion, extra-Poisson, 386, 405
- Distance sampling, 135, 169
- Distribution. *See also* specific distributions
- apparent, 14t, 415

- atlas, 4  
catalog, 49  
importance of, 20  
organism, 4  
study of, 1–6  
true, 14t, 415
- The Distribution and Abundance of Animals*  
(Andrewartha and Birch), 1
- Double-counting, 125
- Dynamic multistate occupancy model, 458–459
- Dynamic occupancy models, 436–450
- Dynamic occurrence and species distribution models, 436–450, 459
- E**
- Ecological process, 8, 392f
- Ecology  
definition of, 1–6  
hierarchical view of, 6, 7f  
metapopulation, 3–4  
state changes in, 2, 2f
- Ecology: Individuals, Populations, and Communities* (Begon), 2
- Ecology: The Experimental Analysis of Distribution and Abundance* (Krebs), 1
- Emigration. *See also* Temporary emigration  
CJS model and, 371  
permanent, 295
- Entry probability, 317, 323, 329, 333, 340, 345
- Errors. *See also* False negatives; False positives  
estimation, 366  
misclassification, 476  
Monte Carlo, 144  
nondetection, 7, 10  
observation, 11  
hierarchical models correcting, 11, 116–117, 117f, 384  
Poisson distribution for, 352  
standard, of estimator, 20
- Estimated population size, 121–126
- Estimation error, bias and, 366
- Estimator, 17, 20
- E-SURGE, 470–471
- Exchangeability, assumption of, 77, 82
- Expected count, 111, 134
- Explanation, objective of, 26
- Explicit models  
defined, 26  
hierarchical, 43, 111, 384, 472  
of misclassification error, 476
- not required, 473  
SECR, 135, 475
- Extra-Poisson dispersion, 386, 405
- Extrinsic nonidentifiability, 216
- F**
- False negatives  
abundance and, 384  
accounting for, 476  
CJS model and, 173  
false positives canceling out, 125  
introduction to, 11–13, 20  
multistate models and, 265  
site-occupancy model and, 417
- False positives  
accounting for, 476  
binomial mixture models and, 388  
CJS model and, 173  
false negatives canceling out, 125  
introduction to, 12–13, 20  
multistate models and, 265  
sample size and, 13  
site-occupancy model and, 416–417
- Fecundity  
integrated population models estimating, 363–366, 377–378
- Poisson GLM modeling, 66–67
- Fidelity, 295
- Finite-population standard deviation, 79
- Fire salamanders, 282–284, 283f
- First breeding model, age-specific  
probability of  
analysis of, 290–294  
description of, 288–289  
generation of simulated data, 289–290  
survival and movement estimation with, 288–294
- First-year fixed observer effect, 105–108
- Fisher Scoring, 58
- Fixed effects  
additive, 154  
in fixed group and random time effect models, 204–208  
in fixed group and time effect models, 199–204  
in fixed group effect models in, 192–194  
in fixed site and fixed year effects model, 100–102  
in fixed site effects model, 99–100  
in fixed time effect models, 184  
random effects compared to, 76, 82

- Fixed effects (*Cont.*)  
 in random site effect, random year effect,  
 and first-year fixed observer effect  
 model, 105–106  
 time effects, 184, 199–204  
 in WinBUGS, 76
- Frailty, 208
- Freeman–Tukey statistic, 224
- Frequentist analysis, of statistical models,  
 28–38
- Freuler, Reto, 126
- G**
- GAM. *See* Generalized additive model
- Gamma prior, 120
- Generalized additive model (GAM)  
 defined, 71  
 development of, 476  
 smoothing of, 131
- Generalized linear mixed model (GLMM)  
 abundance estimation and, 383  
 ANCOVA and, 74–77, 106  
 ANOVA and, 100–106  
 asp viper example, 17–18, 18–19f, 74–77,  
 76f  
 complex estimation in, 95  
 conventional Poisson, for count data  
   analysis of real data, 88–90,  
 95–110  
   fixed site and fixed year effects,  
 100–102  
   fixed site effects, 99–100  
   full model, 108–110  
   generation and analysis of simulated  
 data, 84–88, 92–95  
   introduction to, 73–82  
   overdispersion in, 83  
   with random effects for variability  
 among groups, 90–110  
   random site and random year  
 effects, 103–105  
   random site effect, random year  
 effect, and first-year fixed observer  
 effect, 105–106  
   random site effect, random year  
 effect, first-year fixed observer effect,  
 and overall linear time trend,  
 106–108  
   random site effects, 102–103  
   uses of, 111  
 defined, 71  
 GLM compared to, 88
- introduction to, 73–82  
 latent effects and, 73  
 null, 98–99  
 peregrine falcon example of, 84–90, 86f,  
 92–95  
 summary of, 110–112  
 Swiss coal tit example of, 95f, 95–110,  
 97f
- Generalized linear models (GLM)  
 advantages of, 54  
 ANCOVA, 50, 52  
 binomial  
   analysis of real data, 70–71  
   generation and analysis of simulated  
 data, 68–70  
   for modeling-bounded counts or  
 proportions, 67–71  
   Poisson GLM compared to, 48  
   summary of, 71–72  
   WinBUGS and, 68–71  
 capture–recapture models and, 138  
 effects formulation, 199  
 GLMM compared to, 88  
 introduction to, 48  
 movement among three sites model  
   and, 301  
 peregrine falcon example of, 56–71,  
 56–57f, 65f
- Poisson  
   analysis of real data, 64–66  
   binomial GLM compared to, 48  
   fecundity modeling by, 66–67  
   generation and analysis of  
 simulated data, 56–64  
   overdispersion in, 83  
   in R, 55–66  
   summary of, 71–72  
   WinBUGS and, 55–66  
 random effects in, 73  
 in recruitment, survival, and population  
 size estimation, 328, 345  
 response components in  
   link function and, 54–55  
   noise, 48–55  
   signal, 48–55  
 summary of, 71–72  
 in survival estimation using CJS model,  
 199, 211, 237  
 themes in, 15  
 with WinBUGS, 48  
   binomial GLM and, 68–71  
   Poisson GLM and, 55–66

- time series of counts modeled with, 55–66  
undefined real result trap in, 60–61
- Gibbs sampling, 38
- GLM. *See* Generalized linear models
- GLMM. *See* Generalized linear mixed model
- Goodness of fit (GOF)  
for integrated population models, 380  
m-array and, 222, 224  
in occurrence and species distribution estimation, 418
- Grey-headed woodpecker, 335–339, 336f
- Group effects, 138  
fixed  
in fixed group and random time effect models, 204–208  
in fixed group and time effect models, 199–204  
in fixed group effect models in, 192–194  
individual, 195–199  
models with, 199–208  
random, 194–195
- H**
- Heterogeneity model, 150–153, 335–339
- Hierarchical models. *See also specific models*  
Bayesian statistical modeling and, 43–44  
BUGS language translation of, 25  
defined, 91  
explicit, 43, 111, 384, 472  
implicit, 43, 111, 384, 472  
importance of, 8  
N-mixture, 44  
observation error corrected by, 11, 116–117, 117f, 384  
partitioning in, 13  
power of, 464–472  
for cleaner thinking, 467  
for E-SURGE, 470–471  
for fitting of complex statistical models, 464–465  
for MARK, 470–471  
for PRESENCE, 470–471  
for primary model selection, 468–469  
for secondary model selection, 469–470  
for step-up approach to problems, 467  
for study design, 471–472
- for synthetic understanding of models, 466t, 465–467  
for unmarked, 470–471
- random variables in, 43  
stochastic parts of, 27  
systematic parts of, 27  
variable selection and, 469–470
- Hierarchical scales of organization, 2  
ecology and, 6, 7f  
modeling of, 5  
Scale 1, 2, 3t  
Scale 2, 3, 3t  
Scale 3, 3t, 5  
Scale 4, 3t, 5
- Home range, 135
- Hoopoe, 371f, 371–379, 378f
- House martin, 126–130, 126f, 130f
- I**
- Ibex population, 118–121, 118f, 121f, 124f
- Identifiability, 216–220, 339–341
- Immigration, 371, 377
- Implicit models  
defined, 26  
hierarchical, 43, 111, 384, 472
- Incidence. *See also* Occupancy  
defined, 437  
in metapopulation, 3
- Independence assumption, of integrated population models, 355
- Individual capture heterogeneity models, 335–339
- Individual covariate models  
capture–recapture, 162–168  
for population size estimation, 166–168  
for species richness estimation, 163–166
- Individual effects, 136, 138  
in generation and analysis of simulated data with data augmentation, 150–153  
group, 195–199
- Individual variation, models with  
with fixed group effects, 192–194  
with individual group effects, 195–199  
with random group effects, 194–195  
survival estimation and, 192–199
- Inference, scope of, 78–79
- Information combination  
with Bayes rule, 35  
development of, 474  
as motivation for random-effects model, 81

- Inits function, 128
- Integrated population models
- capture–recapture data in, 357–363
  - count data in, 357–363
  - defined, 348
  - demographic rates, population size, and projection matrices estimation using analysis of, 358–363
  - CJS model in, 371
  - fecundity estimated with, 363–366, 377–378
  - hoopoe example of, 371f, 371–379, 378f
  - introduction to, 348–350
  - ortolan bunting example of, 349, 350–351f, 357–363, 366–370
  - for population viability analysis, 366–370
  - without productivity data, 363–366
  - real data example of, 371–379
  - simple example of, 357–363
  - summary of, 379–380
  - WinBUGS and, 357–379
- development of, 349
- first step, 350–352
  - second step, 352–354
  - third step, 354–357
- fecundity estimated by, 363–366, 377–378
- goodness of fit and, 380
- independence assumption of, 355
- for population viability analysis, 366–370
- reproduction in, 357–363
- simulation with, 349
- state-space likelihood and, 354
- Intrinsic nonidentifiability, 216
- Inverse Wishart distribution, 206–207
- J**
- Joint analysis, of capture–recapture and mark-recovery data
- generation of simulated data, 296–297
  - model for, 295–300
  - survival and movement estimation from, 295–300
- Joint likelihood, formulation of, 354–357
- Jolly–Seber (JS) model
- CJS model compared to, 237, 345
  - data augmentation and, 319–328, 338
- as multistate model, 322–325
- as restricted dynamic occupancy model, 320–322
- superpopulation parameterization and, 325–328
- inferences of, 15
- as multistate model, 322–325, 332
- N-mixture, 475
- observation process in, 318, 318f
- recruitment, survival, and population size estimation using
- analysis of real data, 341–345
  - assumptions of, 316
  - black grouse example of, 329, 329f, 334f
  - connections between parameters in, 339–341
  - data augmentation in, 319–328, 338
  - GLM and, 328, 345
  - grey-headed woodpecker example of, 335–339, 336f
  - identifiability in, 339–341
  - introduction to, 316–317
  - Leisler’s bats example of, 341–345, 344f
  - models with constant survival and time-dependent entry, 328–335
  - models with individual capture heterogeneity, 335–339
  - parameterizations of, 317, 325–328
  - priors in, 326
  - summary of, 345–346
  - WinBUGS and, 321–339, 341–345
- as restricted occupancy model, 331
- state process in, 318, 318f
- as state-space model, 317–319
- superpopulation formulation and, 333
- uses of, 170
- variants, 317
- K**
- Kittiwakes, 414
- Krebs, C. J., 1
- L**
- Lack-of-fit ratio, 401
- Latent abundance, 384
- Latent effects, GLMMs and, 73
- Latent state variable, 181–183, 247, 276, 318
- Latent suitability indicators, 402
- Learning, formalized, 34–35

- Leisler's bats  
 in recruitment, survival, and population size estimation using JS, 341–345, 344f  
 in survival estimation using CJS model, 176f, 231–237, 232f, 235–236f
- Leslie-matrix modeling, 14t, 362, 379
- Levels, 78
- Likelihood. *See also* Maximum likelihood estimate; Multinomial likelihood  
 BUGS language defining, 358  
 of capture–recapture data, 354  
 of count data, 352–354  
 function, 29, 33, 36  
 joint, 354–357  
 principle, 223  
 of reproductive success data, 354  
 state-space  
   integrated population models and, 354  
   in survival and movement estimation using multistate models, 274
- Linear models. *See also* specific models  
 mean structure described by, 49  
 themes in, 15
- Linear predictor, for detection, 386
- Link function, 54–55, 422
- Little owls, 177–183, 178f, 185, 189, 196, 209, 227–231
- Little ringed plover, 270–274, 271f, 289–290, 302–303
- Log-function, 386, 394f
- Logistic normal, 150
- Logit, 148, 154, 186, 430
- Logit-link function, 386, 394f
- Long-eared owl, 439–445, 440f, 454, 456
- M**
- Magden, 126–130
- Marginal probability, 33
- MARK, 470–471
- Markov chain Monte Carlo (MCMC)  
 benefits of, 358  
 development of, 38  
 mark-recovery data and, 261  
 random effects and, 406  
 in restricted occupancy model, 343  
 sampling techniques, 38  
 state-space models and, 120, 130  
 WinBUGS blackbox of, 38–41, 44
- Mark-recovery data  
 capture–recapture data compared to, 295  
 joint analysis of  
   generation of simulated data, 296–297  
   model for, 295–300  
   survival and movement estimation from, 295–300  
 simulations with, 244–246, 296–297  
 survival estimation using  
   common terns example of, 244–246, 245f, 252–255  
   introduction to, 241–242  
   latent state variable in, 247  
   m-array for, 249, 251–253, 255  
   MCMC and, 261  
   multinomial likelihood and, 248–255, 261  
   priors in, 258, 260  
   R code and, 249  
   real-data example of, 255–261  
   red kites example, 255–261, 256f, 259–260f  
   sampling for, 242  
   simulation of, 244–246  
   state-space models and, 242–248, 261  
   summary of, 261  
   WinBUGS and, 244–261
- Mark-recovery models  
 CJS model compared to, 243–244, 248  
 multinomial likelihood fitting, 248–255  
   model with age-dependent parameters, 252–255  
   model with constant parameters, 248–252  
 as state-space model  
   advantages of, 242–248, 261  
   analysis of model with constant parameters, 246–248  
   simulation of mark-recovery data, 244–246  
   state and observation processes, 243f, 243
- M-array  
 CJS model fitted via, 44, 220–231  
 goodness of fit and, 222–224  
 mark-recovery data and, 249, 251–253, 255
- Maximum likelihood estimate (MLE)  
 benefits of, 31  
 from data cloning, 36

- Maximum likelihood estimate (MLE) (*Cont.*)  
 defined, 29  
 mode and, 28, 37  
 posterior mean and, 28  
 tadpole example of, 29, 30f, 33
- MCMC. *See* Markov chain Monte Carlo
- Mean effects, 78
- Memory, 160
- Metacommunity, 3t, 5
- Metapopulation, 15, 96  
 ecology, 3–4  
 extent of, 4  
 incidence in, 3
- Metapopulation designs  
 abundance estimation and, 15, 384  
 analysis of real data, 396–409  
 covariates in, 390–396  
 ecological process in, 392f  
 generation and analysis of simulated data, 388–396  
 GLMM and, 383  
 introduction to, 383–388  
 N-mixture model with  
 overdispersion in, 404–409  
 observation process in, 392f  
 open-population N-mixture model in, 396–409  
 posterior distributions in, 394f, 397, 401f  
 program R in, 396  
*p*-values in, 397  
 silver-washed fritillary example of, 396–409, 397f, 408f  
 simple Poisson model for, 398–401  
 simplest case, 388–390  
 summary of, 409–410  
 WinBUGS and, 388–410  
 ZIP N-mixture model, 401–404
- community models for, 474–475  
 in occurrence and species distribution estimation  
 analysis of real data, 427–436, 445–450  
 dynamic models for, 436–450, 459  
 generation and analysis of simulated data in, 420–427, 439–445  
 goodness of fit and, 418  
 introduction to, 414–418  
 long-eared owl example of, 439–445, 440f, 454, 456  
 multistate occupancy models, 450–458
- p*-values and, 419–420  
*Rosalia alpina* example of, 427–436, 428–429f, 432f, 434f  
 single-season occupancy analysis, 420–427  
 six-spot burnet example of, 445–450, 445f, 450f  
 summary of, 459–460  
 WinBUGS in, 419–436, 439–450  
 population models for, 474–475
- Misclassification error, 476
- Mixed models. *See* Generalized linear mixed model
- MLE. *See* Maximum likelihood estimate
- Mode, 28, 37
- Modeling. *See also specific types of modeling*  
 in algebra, 25, 267  
 flexible, of covariates, 476  
 of hierarchical scales of organization, 5  
 objectives, 26  
 of parameter correlations, 79–80  
 role of, 24–27  
 sayings about, 24  
 summary of, 44–45  
 synthetic understanding of, 465–467, 466t  
 themes, 15
- Modeling-bounded counts or proportions  
 analysis of real data, 70–71  
 binomial GLM for, 67–71  
 generation and analysis of simulated data, 68–70
- Models. *See also specific models*  
 $M_b$ , 148–150  
 $M_h$ , 150–153  
 $M_t$ , 145–147  
 $M_{tbh}$ , 157–162  
 $M_{th}$ , 154–157  
 $M_{t+x}$   
 capture–recapture data and, 162–168  
 for population size estimation, 162–168  
 for species richness, 163–166
- selection of, 37  
 detection probability and, 473  
 primary, 468–469  
 secondary, 469–470
- Monte Carlo error, 144
- Movement among three sites, model of  
 analysis of, 304–306  
 description of, 300–302  
 generation of simulated data, 302–303

- GLM and, 301  
survival and movement estimation using, 300–306
- Movement and survival estimation, from capture–recapture data using multistate models  
age-specific probability of first breeding model, 288–294  
fire salamanders example of, 282–284, 283f  
introduction to, 264–268  
little ringed plover example of, 270–274, 271f, 289–290, 302–303  
model for joint analysis of capture–recapture and mark–recovery data, 295–300  
movement among three sites, 300–306  
movement between two sites, 268–281  
real-data example, 307–311  
showy lady’s slipper example, 307f, 307–311  
state-space likelihood and, 274  
summary of, 311–312  
temporal effects and, 280  
temporary emigration model, 281–287  
WinBUGS and, 270–287, 289–294, 296–300, 302–312
- Movement between two sites, estimated analysis of, 274–281  
described, 268–270  
generation of simulated data, 270–274  
using multistate models, 268–281
- Movement probability, 14t
- Multidimensional arrays, in WinBUGS, 396
- Multievent models, 312
- Multinomial likelihood  
age-dependent models, 227–231, 252–255  
CJS model fitted using, 173, 175, 220–231, 237  
constant parameters and, 248–252  
introduction to, 220–222  
mark-recovery models fitted with, 248–255  
with age-dependent parameters, 252–255  
with constant parameters, 248–252  
survival estimation and, 220–231, 237, 248–255, 261  
time-dependent models, 222–227
- Multiseason site-occupancy models, 436–450
- Multistate models  
algebraic description of, 267  
applications of, 265  
CJS models compared to, 267, 278  
as constant survival and time-dependent entry models, 332  
false negatives and, 265  
false positives and, 265  
JS model as, 322–325, 332  
observation process and, 265, 266f  
occupancy  
dynamic, 458, 459  
extensions of, 458, 459  
importance of, 459  
in occurrence and species distribution estimation, 450–458  
parameterizations of, 453  
uses of, 451  
state process and, 265, 266f  
in survival and movement estimation  
from capture–recapture data  
age-specific probability of first breeding model, 288–294  
fire salamanders example of, 282–284, 283f  
introduction to, 264–268  
little ringed plover example of, 270–274, 271f, 289–290, 302–303  
model for joint analysis of capture–recapture and mark–recovery data, 295–300  
movement among three sites, 300–306  
movement between two sites, 268–281  
real-data example, 307–311  
showy lady’s slipper example, 307f, 307–311  
state-space likelihood and, 274  
summary of, 311–312  
temporal effects and, 280  
temporary emigration model, 281–287  
WinBUGS and, 270–287, 289–294, 296–300, 302–312
- Mutually exclusive events, 32, 32t
- N**
- Negative binomial distribution, 12, 386
- N-mixture model  
basic, 387  
covariates in, 390–396

- N-mixture model (*Cont.*)  
 generalization of, 410  
 hierarchical, 44  
 JS, 475  
 open-population  
   in abundance estimation, 396–409  
   simple Poisson model, 398–401  
 with overdispersion in abundance and detection, 404–409  
 power of, 409  
 simplest case of, 388–390  
 ZIP, 401–404
- Noise  
 link function and, 54–55  
 as response component, 48–55
- Nondetection data. *See* Detection/nondetection data
- Nondetection error, 7, 10
- Nonidentifiability, parameter, 216–217
- Nonparametric assumptions, 476
- Nuisance parameter, 320
- Null model, 98–99
- O**
- Observation  
 equations, 116  
 error  
   hierarchical models correcting, 11, 116–117, 117f, 384  
   Poisson distribution for, 352  
 genesis of, 6–9  
 process, 7f  
   in abundance estimation, 392f  
   accounted for, 13  
   binomial distribution and, 9–13, 20  
   importance of, 472–474  
   in JS models, 318, 318f  
   in mark-recovery models, 243f, 243  
   in multistate models, 265, 266f  
   population analysis and, 8  
   systematic bias in, 121–126  
 true state and, 8  
 variance, 123
- Observed population size, 121–126
- Occam’s razor, 24
- Occupancy. *See also* Site-occupancy models  
 defined, 437  
 as state variable, 4
- Occurrence and species distribution  
 estimation, using site-occupancy models  
 analysis of real data, 427–436, 445–450
- dynamic models for, 436–450, 459  
 generation and analysis of simulated data in, 420–427, 439–445  
 goodness of fit and, 418  
 introduction to, 414–418  
 long-eared owl example of, 439–445, 440f, 454, 456  
 multistate occupancy models, 450–458  
 p-values and, 419–420  
*Rosalia alpina* example of, 427–436, 428–429f, 432f, 434f  
 single-season occupancy analysis, 420–427  
 six-spot burnet example of, 445–450, 445f, 450f  
 summary of, 459–460  
 WinBUGS in, 419–436, 439–450
- OpenBUGS, 390, 439
- Open-population model  
 N-mixture  
   in abundance estimation, 396–409  
   simple Poisson model, 398–401  
 understanding of, 465
- Organism distribution, 4
- Ortolan buntings, 349, 350–351f, 357–363, 366–370
- Overall linear time trend model, 106–108
- Overdispersion  
 capture–recapture data subject to, 196  
 correction for, 386, 404  
 defined, 82  
 N-mixture model with, 404–409  
 in Poisson GLM, 83  
 in Poisson GLMM, 83  
 in random-effects models, 82–90
- Overparameterization, 100
- P**
- Parameterization  
 of JS recruitment, survival, and population size modeling, 317, 325–328  
 of multistate occupancy models, 450–458  
 overparameterization and, 100  
 superpopulation, 325–328  
 treatment contrast, 53
- Parameters  
 abundance, 387  
 age-dependent, 252–255  
 connections between, 339–341  
 constant, 177–183, 246–252  
 correlations, modeling of, 79–80

- defined, 20  
demographic, 373  
estimation and inference, 29  
fixed, 28–29  
identifiability, 216–220, 339–341  
nonidentifiability, 216–217  
nuisance, 320  
of Poisson distribution, 55  
variance, 37
- Pen shell, 166–168, 167f
- Peregrine falcons  
  GLM and, 56–71, 56–57f, 65f  
  GLMM and, 84–90, 86f, 92–95
- Permanent emigration, 295
- Poisson distribution, 12  
  for abundance, 385  
  for observation error, 352  
  parameter of, 55  
  zero-inflated, 386
- Poisson means, 63
- Poisson models  
  for abundance estimation, 398–401  
  GLM  
    analysis of real data, 64–66  
    binomial GLM compared to, 48  
    fecundity modeling by, 66–67  
    generation and analysis of simulated data, 56–64  
    overdispersion in, 83  
    in R, 55–66  
    summary of, 71–72  
    WinBUGS and, 55–66
- GLMM  
  analysis of real data, 88–90, 95–110  
  fixed site and fixed year effects, 100–102  
  fixed site effects, 99–100  
  full model, 108–110  
  generation and analysis of simulated data, 84–88, 92–95  
  introduction to, 73–82  
  overdispersion in, 83  
  with random effects for variability among groups, 90–110  
  random site and random year effects, 103–105  
  random site effect, random year effect, and first-year fixed observer effect, 105–106  
  random site effect, random year effect, first-year fixed observer effect, and overall linear time trend, 106–108
- random site effects, 102–103  
  uses of, 111
- PLN, 83–84
- Poisson-binomial mixture model, 385–386
- Poisson random variable, 55
- Poisson response, 15
- Poisson–log-normal (PLN) model, 83–84
- Population(s). *See also* Integrated population models; Metapopulation; Open-population model; Population size; Superpopulation  
abundance as key descriptor of, 409–410  
analysis  
  for conservation, 477–478  
  defined, 6  
  methods for, 15  
  observation process and, 8  
  recurring themes in, 477  
  viability, 366–370  
  for wildlife management, 477–478
- asp viper, 17–18, 18–19f, 74–77, 76f
- black grouse, 329, 329f, 334f
- classic, 4
- common terns, 244–246, 245f, 252–255
- community as, 5
- counts, state-space models for  
  analysis of simulated data, 119  
  framework of, 117  
  house martin example, 126–130, 126f, 130f  
  ibex example, 118f, 118–121, 121f, 124f  
  introduction to, 115–118  
  MCMC and, 120, 130  
  observation equations in, 116  
  simple, 118–121, 131  
  state-process equations in, 116  
  systematic bias and, 121–126  
  WinBUGS and, 118–130
- descriptors, 7
- in finite-population standard deviation, 79
- fire salamanders, 282–284, 283f
- grey-headed woodpecker, 335–339, 336f
- hoopoe, 371f, 371–379, 378f
- kittiwakes, 414
- Leisler's bats  
  in recruitment, survival, and population size estimation using JS model, 341–345, 344f  
  survival estimation using CJS model and, 231–237, 232f, 235–236f

- Population(s) (*Cont.*)
- likelihood of count data, 352–354
  - little owls, 177–183, 178f, 185, 189, 196, 209, 227–231
  - little ringed plover, 270–274, 271f, 289–290, 302–303
  - long-eared owl, 439–445, 440f, 454, 456
  - models, for metapopulation designs, 474–475
  - ortolan bunting, 349, 350–351f, 357–363, 366–370
  - pen shell, 166–168, 167f
  - peregrine falcon
    - GLM model of, 56–71, 56–57f, 65f
    - GLMM model of, 84–90, 86f, 92–95  - quantities, sample size and, 20
  - red kites, 255–261, 256f, 259–260f
  - Rosalia alpina* beetle, 427–436, 428–429f, 432f, 434f
  - silver-washed fritillary, 396–409, 397f, 408f
  - six-spot burnet, 445–450, 445f, 450f
  - sparrow, 9–11, 11f
  - spotted owl, 414
  - Swiss coal tit, 95f, 95–110, 97f
  - tadpole, 29, 30f, 33, 38
  - unobserved, 366
  - white storks, 252
  - wryneck, 156f
- Population size
- closed, capture–recapture data for estimating
    - analysis of real data, 157–162
    - behavioral effects in, 148–150
    - CJS model and, 135, 137–138
    - combined effects in, 154–157
    - generation and analysis of simulated data, 139–157
    - individual effects in, 150–153
    - introduction to, 134–139
    - model  $M_{tbh}$  for species richness estimation, 157–162
    - model  $M_{t+u}$ , 162–168
    - pen shell example of, 166–168, 167f
    - summary of, 169–170
    - time effects in, 145–147
    - trap response example of, 136–137, 148–157  - demographic rates and, 350–352
  - individual covariate models estimating, 166–168
  - integrated population models estimating analysis of, 358–363
- CJS model in, 371
- fecundity estimated with, 363–366, 377–378
- hoopoe example of, 371f, 371–379, 378f
- introduction to, 348–350
- ortolan bunting example of, 349, 350–351f, 357–363, 366–370
- for population viability analysis, 366–370
- without productivity data, 363–366
- real data example of, 371–379
- simple example of, 357–363
- summary of, 379–380
- WinBUGS and, 357–379
- JS model estimating
- analysis of real data, 341–345
  - assumptions of, 316
  - black grouse example of, 329, 329f, 334f
  - connections between parameters in, 339–341
  - data augmentation in, 319–328, 338
  - GLM and, 328, 345
  - grey-headed woodpecker example of, 335–339, 336f
  - identifiability in, 339–341
  - introduction to, 316–317
  - Leisler's bats example of, 341–345, 344f
  - models with constant survival and time-dependent entry, 328–335
  - models with individual capture heterogeneity, 335–339
  - parameterizations of, 317, 325–328
  - priors in, 326
  - summary of, 345–346
  - WinBUGS and, 321–339, 341–345
- observed, 121–126
- of occupied patches, 13
- as smoothed index, 380
- true, 121–126
- of unobserved populations, 366
- Positional matching, 87
- Posterior distributions
- in abundance estimation, 394f, 397, 401f
  - introduction to, 33–34, 36–37
- Posterior mean, MLE and, 28
- Posterior sampling, 41–43
- Postwork activity, 32
- Potential individuals, 141
- Precision, 16–20

- Predator control, 477  
Prediction, objective of, 26  
PRESENCE, 470–471  
Presence/absence data, 414, 416. *See also* Detection/nondetection data  
Primary model selection, 468–469  
Priors  
    Bayes rule and, 35–37, 44  
    Dirichlet distribution, 300, 302, 308, 454  
    gamma, 120  
    mark-recovery data and, 258, 260  
    random effects and, 77  
    in recruitment, survival, and population size estimation, 326  
    state-space models and, 127  
    uniform, 120  
    for variance parameters, 37  
Probability  
    age-specific, of first breeding model  
        analysis of, 290–294  
        description of, 288–289  
        generation of simulated data, 289–290  
        survival and movement estimation with, 288–294  
    defined, 28  
detection, 6, 11  
    in binomial mixture models, 387  
    capture–recapture models and, 137t, 136–137  
    constant, 447, 472  
    date related to, 433  
    model selection view of, 473  
    state-space models and, 124–125, 131  
    time of day related to, 433  
    within-capture-history dependence of, 148  
entry, 317, 323, 329, 333, 340, 345  
marginal, 33  
movement, 14t  
recapture, 212–216  
removal entry, 320, 323, 333  
survival, 14t, 15  
Process variance, 123  
Productivity data, estimation without, 363–366  
Program R. *See also* R code  
    in abundance estimation, 396  
    distribution catalog in, 49  
    simulations with, 9, 10
- Projection matrices, integrated population models and analysis of, 358–363  
CJS model in, 371  
fecundity estimated with, 363–366, 377–378  
hoopoe example of, 371f, 371–379, 378f  
introduction to, 348–350  
ortolan bunting example of, 349, 350–351f, 357–363, 366–370  
for population viability analysis, 366–370  
without productivity data, 363–366  
real data example of, 371–379  
reproduction and, 357–363  
summary of, 379–380  
WinBUGS and, 357–379
- Pseudo-individuals, 319, 331
- Pseudoreplication  
    avoidance of, 80  
    in site-occupancy models, 437
- p-values  
    in abundance estimation, 397  
    in Bayesian statistical modeling, 222–223, 226f  
    criticisms of, 222–223  
    in site-occupancy model, 419–420
- Q**
- Quantity  
    defined, 33  
    sample size and, 20
- R**
- R code  
    for ANCOVA linear model, 50, 52  
    mark-recovery data and, 249  
    Poisson GLM in, 55–66  
    presentation of, 16  
    random-effects modeling in, 82–90  
    for survival estimation using CJS model, 178  
    treatment contrast parameterization and, 53
- R function, 63–64, 221
- Random effects  
    annual rates as, 377  
    costs of, 82  
    defined, 73, 77–78  
    fixed effects compared to, 76, 82  
    flexibility from, 15  
    GLM and, 73  
    group, 194–195

- Random effects (*Cont.*)
- in linear predictor for detection, 386
  - MCMC and, 406
  - for overdispersion correction, 404
  - priors and, 77
  - time effects, 184–188, 204–208
  - for variability among groups, 90–110
  - in WinBUGS, 76, 82–90
- Random factor, 78
- Random processes, accounting for, 80
- Random site and random year effects model, 103–105
- Random site effect, random year effect, and first-year fixed observer effect model, 105–106
- Random site effect, random year effect, first-year fixed observer effect, and overall linear time trend model, 106–108
- Random site effects model, 102–103
- Random temporary emigration, 436
- Random time effect models, 184–188
- Random variables, 27
- in fitting of complex statistical models, 464–465
  - in hierarchical models, 43
  - Poisson, 55
- Random-coefficients models, 78
- Random-effects models
- ANCOVA, 75–76
  - data augmentation in, 152
  - difficulty fitting, 191
  - motivations, 78–81
  - accounting for all random processes, 80
  - borrowed strength, 80–81, 110
  - information combination, 81
  - modeling parameter correlations, 79–80
  - partitioning of variability, 79
  - pooling of batched effects, 81
  - pseudoreplication avoidance, 80
  - scope of inference, 78–79
  - variability assessment, 79
- overdispersion accounted for in, 82–90
- in R, 82–90
- random site and random year effects model, 103–105
- random site effect, random year effect, and first-year fixed observer effect model, 105–106
- random site effect, random year effect, first-year fixed observer effect, and overall linear time trend model, 106–108
- random site effects model, 102–103
- random time effect model, 184–188
- terminology, 78
- in WinBUGS, 82–90
- Random-intercepts models, 75, 78
- Range, 4
- Recapture probability, survival estimation and, 212–216
- Recruitment, survival, and population size estimation using JS model
- analysis of real data, 341–345
  - assumptions of, 316
  - black grouse example of, 329, 329f, 334f
  - connections between parameters in, 339–341
  - data augmentation in, 319–328, 338
  - GLM and, 328, 345
  - grey-headed woodpecker example of, 335–339, 336f
  - identifiability in, 339–341
  - introduction to, 316–317
  - Leisler's bats example of, 341–345, 344f
  - models with constant survival and time-dependent entry, 328–335
  - models with individual capture heterogeneity, 335–339
  - parameterizations of, 317, 325–328
  - priors in, 326
  - summary of, 345–346
  - WinBUGS and, 321–339, 341–345
- Recruitment probability. *See* Entry probability
- Red kites, 255–261, 256f, 259–260f
- Red-backed shrikes, 213, 214f
- Regression model, 74
- Reif, Jiri, 157
- REML method, 75
- Removal entry probability, 320, 323, 333
- Replicated counts, 385, 409
- Reproduction, in integrated population model, 357–363
- Reproductive success data, 354
- Response components, in GLM
- link function and, 54–55
  - noise, 48–55
  - signal, 48–55
- Restricted occupancy model
- dynamic, 320–322
  - JS model as, 331

- MCMC samples for, 343  
speed of, 341
- Rhat statistic, 39, 62, 63f, 87
- Robust design, 387, 472
- Rosalia alpina* beetle, 427–436, 428–429f, 432f, 434f
- S**
- Sample size  
false positives and, 13  
population quantities and, 20
- Sampling  
of capture–recapture data, 472  
design, 468  
distance, 135, 169  
Gibbs, 38  
for mark-recovery data, 242  
MCMC, 38  
posterior, 41–43  
repeated, 17  
variation, 13, 19–20, 422
- Scale 1, 2, 3t
- Scale 2, 3, 3t
- Scale 3, 3t, 5
- Scale 4, 3t, 5
- Science, modeling’s role in, 24–27
- Seasons, as grouping factor, 437
- Secondary model selection, 469–470
- SECR models. *See* Spatially explicit capture–recapture models
- Showy lady’s slipper, 307f, 307–311
- Shrinkage, 80, 377
- Signal  
link function and, 54–55  
as response component, 48–55
- Silver-washed fritillary, 396–409, 397f, 408f
- Simulations  
abundance and covariate relationship in, 395f  
in abundance estimation, 388–396  
with age-specific probability of first breeding model, 289–290  
asp viper, 17–18, 18–19f, 74–77, 76f  
benefits of, 16–20  
bias and, 16–20  
binomial GLM and, 68–70  
with capture–recapture models, 139–157  
behavioral effects, 148–150  
combined effects, 154–157  
individual effects, 150–153  
introduction to, 139–145  
time effects, 145–147
- from inside out, 17
- with integrated population models, 349
- of mark-recovery data, 244–246, 296–297
- in model for joint analysis of capture–recapture and mark–recovery data, 296–297
- of movement among three sites, 302–303
- of movement between two sites, 270–274
- in occurrence and species distribution estimation, 420–427, 439–445
- Poisson GLM and, 56–64
- Poisson GLMM and, 84–88, 92–95
- with program R, 9–10
- state-space models and, 119
- in temporary emigration model, 282–284
- Single-season occupancy analysis  
model with covariates, 422–427  
in occurrence and species distribution estimation, 420–427
- Rosalia alpina* beetle example of, 427–436, 428–429f, 432f, 434f
- simplest model for, 420–422
- Site covariate, 390
- Site effects, 90–110
- Site-occupancy models  
assumptions of, 416  
Bernoulli distributions and, 417  
defined, 414  
dynamic, 436–450, 459  
false negatives and, 417  
false positives and, 416–417  
metapopulation designs and, 474–475  
multiseason, 436–450  
multistate  
dynamic, 458–459  
extensions of, 458–459  
importance of, 459  
in occurrence and species distribution estimation, 450–458  
parameterizations of, 453  
uses of, 451
- occurrence and species distribution estimation using  
analysis of real data, 427–436, 445–450  
dynamic models for, 436–450, 459  
generation and analysis of simulated data in, 420–427, 439–445  
goodness of fit and, 418  
introduction to, 414–418

- Site-occupancy models (*Cont.*)
- long-eared owl example of, 439–445, 440f, 454, 456
  - multistate occupancy models, 450–458
  - p*-values and, 419–420
  - Rosalia alpina* example of, 427–436, 428–429f, 432f, 434f
  - single-season occupancy analysis, 420–427
  - six-spot burnet example of, 445–450, 445f, 450f
  - summary of, 459–460
  - WinBUGS in, 419–436, 439–450
  - p* not accounted for in, 419–420
  - pseudoreplication in, 437
  - simplest version of, 420–422
  - for single-season occupancy analysis
    - model with covariates, 422–427
    - in occurrence and species distribution estimation, 420–427
    - Rosalia alpina* beetle example of, 427–436, 428–429f, 432f, 434f
    - simplest model for, 420–422
  - for species distributions, 15, 419–420
  - state process of, 439
  - temporary emigration in, 436–437
- Six-spot burnet, 445–450, 445f, 450f
- Smoothed estimate, 131
- Sparrows, 9–11, 11f
- Spatial models
- development of, 475
  - SECR, 135, 475
  - for study design, 475
- Spatial variation, in abundance, 384
- Spatially explicit capture–recapture (SECR) models, 135, 475
- Species distribution
- estimation
    - analysis of real data, 427–436, 445–450
    - dynamic models for, 436–450, 459
    - generation and analysis of simulated data in, 420–427, 439–445
    - goodness of fit and, 418
    - introduction to, 414–418
    - long-eared owl example of, 439–445, 440f, 454, 456
    - multistate occupancy models, 450–458
    - p*-values and, 419–420
- Rosalia alpina* example of, 427–436, 428–429f, 432f, 434f
- single-season occupancy analysis, 420–427
- six-spot burnet example of, 445–450, 445f, 450f
- summary of, 459–460
- WinBUGS in, 419–436, 439–450
- site-occupancy model for, 15, 419–420
- species–abundance, 5
- Species richness, 139
- biodiversity and, 5
  - model  $M_{t|t_0}$  for estimating, 157–162
  - model  $M_{t+X}$  for estimating, 162–166
- Species–abundance distribution, 5
- Spline modeling, 476
- Spotted owl, 414
- Standard error, of estimator, 20
- State
- changes in, 2, 2f
  - defined, 264
  - observed, 8
  - process
    - equations, 116, 265
    - in JS model, 318, 318f
    - in mark-recovery model, 243f, 243
    - multistate models and, 265, 266f
    - in site-occupancy model, 439
  - true, 8
  - variables, 6
    - abundance, 410, 414
    - latent, 181–183, 247, 276, 318
    - occupancy, 4
- State-space likelihood
- integrated population models and, 354
  - in survival and movement estimation
    - using multistate models, 274
- State-space models
- CJS models and, 173, 175–177, 181, 237, 465, 466f
  - defined, 111, 115
  - detection probability and, 124–125, 131
  - JS model as, 317–319
  - mark-recovery models as
    - advantages of, 242–248, 261
    - analysis of model with constant parameters, 246–248
    - simulation of mark-recovery data, 244–246
    - state and observation processes, 243, 243f

- for population counts  
analysis of simulated data, 119  
framework of, 117  
house martin example, 126–130, 126f, 130f  
ibex example, 118f, 118–121, 121f, 124f  
introduction to, 115–118  
MCMC and, 120, 130  
observation equations in, 116  
simple, 118–121, 131  
state-process equations in, 116  
systematic bias and, 121–126  
WinBUGS and, 118–130
- priors and, 127  
summary of, 131
- survival estimation using mark-recovery  
data and, 242–248, 261
- Statistical models. *See also* Bayesian  
statistical modeling  
complex, fitting of, 464–465  
frequentist analysis of, 28–38  
introduction to, 27  
organic approach to, 469  
response components in, 48–55
- Step-up approach, to problems, 467
- Strength, borrowed, 80–81, 110
- Study design  
hierarchical models for, 471–472  
spatial models for, 475
- Suitability indicators, latent, 402
- Superpopulation  
in constant survival and time-dependent  
entry models, 333  
parameterization, 325–328
- Survival estimation  
using CJS model  
age effect models and, 208–212  
age-dependent models and, 227–231  
analysis of real data, 231–237  
constant parameters and, 177–183  
fixed group and random time effect  
models in, 204–208  
fixed group and time effect models  
in, 199–204  
fixed group effect models in,  
192–194  
fixed time effect models in, 184  
GLM and, 199, 211, 237  
individual group effects in, 195–199  
introduction to, 172–175  
latent state variable in, 181–183
- Leisler's bat example, 231–237, 232f, 235–236f  
little owls example of, 177–183, 178f, 185, 189, 196, 209, 227–231  
models with individual variation  
and, 192–199  
multinomial likelihood and,  
220–231, 237  
parameter identifiability in, 216–220  
R code and, 178  
random group effects in, 194–195  
random time effect models in,  
184–188, 204–208  
recapture probability and, 212–216  
red-backed shrikes example, 213, 214f  
summary of, 237–238  
temporal covariates in, 188–192  
temporal variability, 204  
time and group effect models in,  
199–208  
time-dependent models and, 222–227  
time-variation models and, 183–192  
trap response models and, 212–216  
WinBUGS and, 177–208, 212–237
- using JS model  
analysis of real data, 341–345  
assumptions of, 316  
black grouse example of, 329, 329f, 334f  
connections between parameters in,  
339–341  
data augmentation in, 319–328, 338  
GLM and, 328, 345  
grey-headed woodpecker example  
of, 335–339, 336f  
identifiability in, 339–341  
introduction to, 316–317  
Leisler's bats example of, 341–345,  
344f  
models with constant survival and  
time-dependent entry, 328–335  
models with individual capture  
heterogeneity, 335–339  
parameterizations of, 317, 325–328  
priors in, 326  
summary of, 345–346  
WinBUGS and, 321–339, 341–345
- using mark-recovery data  
common terns example of, 244–246,  
245f, 252–255  
introduction to, 241–242  
latent state variable in, 247

- Survival estimation (*Cont.*)
- m-array for, 249, 251–253, 255
  - MCMC and, 261
  - multinomial likelihood and, 248–255, 261
  - priors in, 258, 260
  - R code and, 249
  - real-data example of, 255–261
  - red kites example, 255–261, 256f, 259–260f
  - sampling for, 242
  - simulation of, 244–246
  - state-space models and, 242–248, 261
  - summary of, 261
  - WinBUGS and, 244–261
- using multistate models
- age-specific probability of first breeding model, 288–294
  - fire salamanders example of, 282–284, 283f
  - introduction to, 264–268
  - little ringed plover example of, 270–274, 271f, 289–290, 302–303
  - model for joint analysis of capture–recapture and mark–recovery data, 295–300
  - movement among three sites, 300–306
  - movement between two sites, 268–281
  - real-data example, 307–311
  - showy lady’s slipper example, 307f, 307–311
  - state-space likelihood and, 274
  - summary of, 311–312
  - temporal effects and, 280
  - temporary emigration model, 281–287
  - WinBUGS and, 270–287, 289–294, 296–300, 302–312
- Survival probability, modeled, 14t, 15
- Swiss coal tit, 95f, 95–110, 97f
- Swiss red kites, 255–261, 256f, 259–260f
- Synthetic understanding, of models, 466t, 465–467
- Systematic bias, state-space models and, 121–126
- T**
- Tadpole example, 29, 30f, 33, 38
- Temporal covariates, models with, 188–192
- Temporal effects, 280
- Temporal variability
- of demographic parameters, 373
  - in survival estimates, 204
- Temporary emigration
- challenge of dealing with, 135
  - closure and, 416
  - model
  - analysis of, 284–287
  - description of, 281–282
  - generation of simulated data, 282–284
  - survival and movement estimation from, 281–287
  - random, 436
  - in site-occupancy models, 436–437
- Territory mapping method, 95
- Three sites, movement among
- analysis of, 304–306
  - description of, 300–302
  - generation of simulated data, 302–303
  - GLM and, 301
  - survival and movement estimation using, 300–306
- Time and group effect models, 199–208
- Time effects, 136, 138
- age effects combined with, 212
  - fixed, 184, 199–204
  - in generation and analysis of simulated data with data augmentation, 145–147
  - random, 184–188, 204–208
  - in time and group effect models, 199–208
- Time series, of counts, 55–66
- Time-dependent models, 223–227. *See also* Constant survival and time-dependent entry models
- Time-variation, models with
- with fixed time effects, 184
  - with random time effects, 184–188
  - survival estimation with, 183–192
  - with temporal covariates, 188–192
- Trap response
- closed population size estimation and, 136–137, 148–157
  - survival estimation and, 212–216
- Trap-happy effect, 137, 213
- Trap-shyness, 137
- Treatment contrast parameterization, 53
- Trend, 5
- True abundance, 14t
- True distribution, 14t, 415
- True population size, 121–126

True state, observed state compared to, 8  
 Two sites, movement between  
     analysis of, 274–281  
     described, 268–270  
     generation of simulated data, 270–274  
     using multistate models, 268–281

**U**

Ultrastructural modeling, 189  
 Unbiased estimate, 17  
 Uncertainty, evaluation of, 28  
 Uniform priors, 120  
 Unmarked, 470–471  
 Unobserved populations, estimates of, 366

**V**

Variability  
     assessment, random-effects models and, 79  
     among groups, 90–110  
     partitioning of, 79  
     temporal, 204, 373  
 Variance  
     decomposition, 8  
     observation, 123  
     parameters, 37  
     process, 123  
 Varying-intercepts models, 78  
 Varying-slopes models, 78  
 Viability analysis, 366–370  
 Vital rates, 14t, 15

**W**

White storks, 252  
 Wildlife management, population analysis  
     for, 477–478  
 WinBUGS. *See also* BUGS language  
     in abundance estimation, 388–410  
     Bayesian statistical modeling and, 38–40  
     capture–recapture models and, 138–169  
     cleaner thinking in, 467  
     fixed effects in, 76  
     flexibility of, 41

GLM with, 48  
     binomial GLM and, 68–71  
     Poisson GLM and, 55–66  
     time series of counts modeled with, 55–66  
     undefined real result trap in, 60–61  
     idiiosyncrasies of, 40  
 integrated population models and, 357–379  
 MARK compared to, 471  
     as MCMC blackbox, 38–41, 44  
     multidimensional arrays in, 396  
     in occurrence and species distribution  
         estimation, 419–436, 439–450  
     presentation of, 16  
 primary model selection in, 468–469  
 random effects in, 76, 82–90  
 in recruitment, survival, and population  
     size estimation using JS model,  
         321–339, 341–345  
 secondary model selection in, 469–470  
 as standalone software, 40  
 state-space models for population counts  
     and, 118–130  
 in step-up approach, 467  
 for survival and movement estimation  
     using multistate models, 270–287,  
         289–294, 296–300, 302–312  
 in survival estimation using CJS model,  
     177–208, 212–237  
 in survival estimation using mark–  
     recovery data, 244–261

Within-capture-history dependence, of  
     detection probability, 148

Wryneck, 158f

**Y**

Year effects, 90–110  
 Yobs, 140

**Z**

Zero-inflated Poisson distribution, 386  
 Zero-inflated Poisson (ZIP) N-mixture  
     model, 401–404