Bayesian Cognitive Modeling: A Practical Course

MICHAEL D. LEE AND ERIC-JAN WAGENMAKERS

Contents

For Colleen and David, and Helen and Mitchell — Michael

Preface

This book, together with the code, answers to questions, and other material at `www.bayesmodels.com`, teaches you how to do Bayesian modeling. Using modern computer software—and, in particular, the WinBUGS program—this turns out to be surprisingly straightforward. After working through the examples provided in this book, you should be able to build your own models, apply them to your own data, and draw your own conclusions.

This book is based on three principles. The first is that of *accessibility*: the book's only prerequisite is that you know how to operate a computer; you do not need any advanced knowledge of statistics or mathematics. The second principle is that of *applicability*: the examples in this book are meant to illustrate how Bayesian modeling can be useful for problems that people in cognitive science care about. The third principle is that of *practicality*: this book offers a hands-on, "just do it" approach that we feel keeps students interested and motivated.

In line with these three principles, this book has little content that is purely theoretical. Hence, you will not learn from this book why the Bayesian philosophy to inference is as compelling as it is; neither will you learn much about the intricate details of modern sampling algorithms such as Markov chain Monte Carlo, even though this book could not exist without them.

The goal of this book is to facilitate and promote the use of Bayesian modeling in cognitive science. As shown by means of examples throughout this book, Bayesian modeling is ideally suited for applications in cognitive science. It is easy to construct a basic model, and then add individual differences, add substantive prior information, add covariates, add a contaminant process, and so on. Bayesian modeling is flexible and respects the complexities that are inherent in the modeling of cognitive phenomena.

We hope that after completing this book, you will have gained not only a new understanding of statistics (yes, it can make sense), but also the technical skills to implement statistical models that professional but non-Bayesian cognitive scientists dare only dream about.

Michael D. Lee
Irvine, USA
Eric-Jan Wagenmakers
Amsterdam, The Netherlands

Acknowledgements

The plan to produce this book was hatched in 2006. Since then, the core material has undergone a steady stream of additions and revisions. The revisions were inspired in part by students and colleagues who relentlessly suggested improvements, pointed out mistakes, and attended us to inconsistencies and inefficiencies. We would especially like to thank Ryan Bennett, Adrian Brasoveanu, Eddy Davelaar, Joram van Driel, Wouter Kruijne, Alexander Ly, John Miyamoto, James Negen, Thomas Palmeri, James Pooley, Don van Ravenzwaaij, Hedderik van Rijn, J. P. de Ruiter, Anja Sommavilla, Helen Steingroever, Wolf Vanpaemel, and Ruud Wetzels for their constructive comments and contributions. We are particularly grateful to Dora Matzke for her help in programming and plotting. Any remaining mistakes are the sole responsibility of the authors. A list of corrections and typographical errors will be available on `www.bayesmodels.com`. When you spot a mistake or omission that is not on the list please do not hesitate to email us at `BayesModels@gmail.com`.

The material in this book is not independent of our publications in the cognitive science literature. Sometimes, an article was turned into a book chapter; at other times, a book chapter spawned an article. Here we would like to acknowledge our published articles that contain text and figures resembling, to varying degrees, those used in this book. These articles often may be consulted for a more extensive and formal exposition of the material at hand.

Chapter 1: The basics of Bayesian analysis

- Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method. *Cognitive Psychology, 60*, 158–189.

Chapter 6: Latent-mixture models

- Ortega, A., Wagenmakers, E.-J., Lee, M. D., Markowitsch, H. J., & Piefke, M. (2012). A Bayesian latent group analysis for detecting poor effort in the assessment of malingering. *Archives of Clinical Neuropsychology, 27*, 453–465.

Chapter 7: Bayesian model comparison

- Scheibehenne, B., Rieskamp, J., & Wagenmakers, E.-J. (2013). Testing adaptive toolbox models: A Bayesian hierarchical approach. *Psychological Review, 120,* 39–64.

- Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage–Dickey method. *Cognitive Psychology, 60*, 158–189.

Chapter 8: Comparing Gaussian means

- Wetzels, R., Raaijmakers, J. G. W., Jakab, E., & Wagenmakers, E.-J. (2009). How to quantify support for and against the null hypothesis: A flexible WinBUGS implementation of a default Bayesian t test. *Psychonomic Bulletin & Review, 16*, 752–760.

Chapter 9: Comparing binomial rates

- Wagenmakers, E.-J., Lodewyckx, T., Kuriyal, H., & Grasman, R. (2010). Bayesian hypothesis testing for psychologists: A tutorial on the Savage-Dickey method. *Cognitive Psychology, 60*, 158–189.

Chapter 10: Memory retention

- Shiffrin, R. M., Lee, M. D., Kim, W., & Wagenmakers, E.-J. (2008). A survey of model evaluation approaches with a tutorial on hierarchical Bayesian methods. *Cognitive Science, 32*, 1248–1284.

Chapter 11: Signal detection theory

- Lee, M. D. (2008). BayesSDT: Software for Bayesian inference with signal detection theory. *Behavior Research Methods, 40*, 450–456.
- Lee, M. D. (2008). Three case studies in the Bayesian analysis of cognitive models. *Psychonomic Bulletin & Review, 15*, 1–15.

Chapter 13: Extrasensory perception

- Wagenmakers, E.-J. (2012). Can people look into the future? Contribution in honor of the University of Amsterdam's 76th lustrum.
- Wagenmakers, E.-J., Wetzels, R., Borsboom, D., van der Maas, H. L. J., & Kievit, R. A. (2012). An agenda for purely confirmatory research. *Perspectives on Psychological Science, 7*, 627–633.

Chapter 14: Multinomial processing trees

- Matzke, D., Dolan, C. V., Batchelder, W. H., & Wagenmakers, E.-J. (in press). Bayesian estimation of multinomial processing tree models with heterogeneity in participants and items. *Psychometrika*.

Chapter 15: The SIMPLE model of memory

- Shiffrin, R. M., Lee, M. D., Kim, W., & Wagenmakers, E.-J. (2008). A survey of model evaluation approaches with a tutorial on hierarchical Bayesian methods. *Cognitive Science, 32*, 1248–1284.

Chapter 16: The BART model of risk taking

- van Ravenzwaaij, D., Dutilh, G., & Wagenmakers, E.-J. (2011). Cognitive model decomposition of the BART: Assessment and application. *Journal of Mathematical Psychology, 55*, 94–105.

Chapter 17: Generalized context model

- Lee, M. D. & Wetzels, R. (2010). Individual differences in attention during category learning. In R. Catrambone & S. Ohlsson (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pp. 387–392. Austin, TX: Cognitive Science Society.
- Bartlema, A., Lee, M. D., Wetzels, R., & Vanpaemel, W. (2012). Bayesian hierarchical mixture models of individual differences in selective attention and representation in category learning. Manuscript submitted for publication.

Chapter 18: Heuristic decision-making

- Lee, M. D. & Newell, B. R. (2011). Using hierarchical Bayesian methods to examine the tools of decision-making. *Judgment and Decision Making, 6*, 832–842.

Chapter 19: Number concept development

- Lee, M. D. & Sarnecka, B. W. (2010). A model of knower-level behavior in number-concept development. *Cognitive Science, 34*, 51–67.
- Lee, M. D. & Sarnecka, B. W. (2011). Number knower-levels in young children: Insights from a Bayesian model. *Cognition, 120*, 391–402.

PART I

GETTING STARTED

[T]he theory of probabilities is basically just common sense reduced to calculus; it makes one appreciate with exactness that which accurate minds feel with a sort of instinct, often without being able to account for it.

Laplace, 1829

1 The basics of Bayesian analysis

1.1 General principles

The general principles of Bayesian analysis are easy to understand. First, uncertainty or "degree of belief" is quantified by probability. Second, the observed data are used to update the *prior* information or beliefs to become *posterior* information or beliefs. That's it!

To see how this works in practice, consider the following example. Assume you are given a test that consists of 10 factual questions of equal difficulty. What we want to estimate is your ability, which we define as the rate θ with which you answer questions correctly. We cannot directly observe your ability θ. All that we can observe is your score on the test.

Before we do anything else (for example, before we start to look at your data) we need to specify our prior uncertainty with respect to your ability θ. This uncertainty needs to be expressed as a probability distribution, called the *prior distribution*. In this case, keep in mind that θ can range from 0 to 1, and that we do not know anything about your familiarity with the topic or about the difficulty level of the questions. Then, a reasonable "prior distribution," denoted by $p(\theta)$, is one that assigns equal probability to every value of θ. This uniform distribution is shown by the dotted horizontal line in Figure 1.1.

Now we consider your performance, and find that you answered 9 out of 10 questions correctly. After having seen these data, the updated knowledge about θ is described by the *posterior distribution*, denoted $p(\theta \mid D)$, where D indicates the observed data. This distribution expresses the uncertainty about the value of θ, quantifying the relative probability that each possible value is the true value. Bayes' rule specifies how we can combine the information from the data—that is, the likelihood $p(D \mid \theta)$—with the information from the prior distribution $p(\theta)$, to arrive at the posterior distribution $p(\theta \mid D)$:

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)}. \tag{1.1}$$

This equation is often verbalized as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}. \tag{1.2}$$

Note that the marginal likelihood (i.e., the probability of the observed data) does not involve the parameter θ, and is given by a single number that ensures that

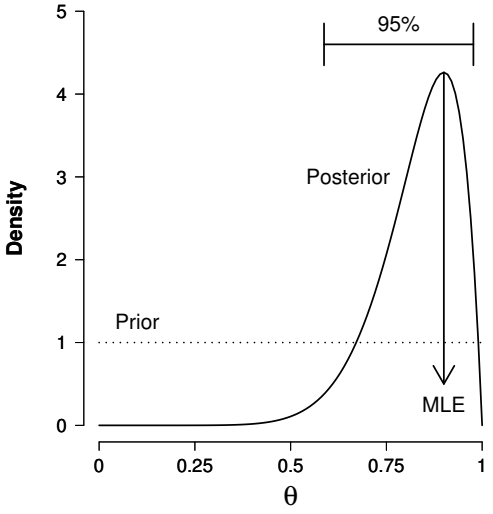


Fig. 1.1 Bayesian parameter estimation for rate parameter θ, after observing 9 correct responses and 1 incorrect response. The mode of the posterior distribution for θ is 0.9, equal to the maximum likelihood estimate (MLE), and the 95% credible interval extends from 0.59 to 0.98.

the area under the posterior distribution equals 1. Therefore, Equation 1.1 is often written as

$$p(\theta \mid D) \propto p(D \mid \theta)\, p(\theta), \tag{1.3}$$

which says that the posterior is proportional to the likelihood times the prior. Note that the posterior distribution is a combination of what we knew before we saw the data (i.e., the information in the prior distribution), and what we have learned from the data. In particular, note that the new information provided by the data has reduced our uncertainty about the value of θ, as shown by the posterior distribution being narrower than the prior distribution.

The solid line in Figure 1.1 shows the posterior distribution for θ, obtained when the uniform prior is updated with the data. The central tendency of a posterior distribution is often summarized by its mean, median, or mode. Note that with a uniform prior, the mode of a posterior distribution coincides with the classical maximum likelihood estimate or MLE , $\hat{\theta} = k/n = 0.9$ (Myung, 2003). The spread of a posterior distribution is most easily captured by a Bayesian x% credible interval that extends from the $(100 - x)/2^{\text{th}}$ to the $(100 + x)/2^{\text{th}}$ percentile of the posterior distribution. For the posterior distribution in Figure 1.1, a 95% Bayesian credible interval for θ extends from 0.59 to 0.98. In contrast to the orthodox confidence interval, this means that one can be 95% confident that the true value of θ lies in between 0.59 and 0.98.

Exercises

Exercise 1.1.1 The famous Bayesian statistician Bruno de Finetti published two big volumes entitled *Theory of Probability* (de Finetti, 1974). Perhaps surprisingly, the first volume starts with the words "probability does not exist." To understand why de Finetti wrote this, consider the following situation: someone tosses a fair coin, and the outcome will be either heads or tails. What do you think the probability is that the coin lands heads up? Now suppose you are a physicist with advanced measurement tools, and you can establish relatively precisely both the position of the coin and the tension in the muscles immediately before the coin is tossed in the air—does this change your probability? Now suppose you can briefly look into the future (Bem, 2011), albeit hazily. Is your probability still the same?

Exercise 1.1.2 On his blog, prominent Bayesian Andrew Gelman wrote (March 18, 2010): "Some probabilities are more objective than others. The probability that the die sitting in front of me now will come up '6' if I roll it ... that's about 1/6. But not exactly, because it's not a perfectly symmetric die. The probability that I'll be stopped by exactly three traffic lights on the way to school tomorrow morning: that's well, I don't know exactly, but it is what it is." Was de Finetti wrong, and is there only one clearly defined probability of Andrew Gelman encountering three traffic lights on the way to school tomorrow morning?

Exercise 1.1.3 Figure 1.1 shows that the 95% Bayesian credible interval for θ extends from 0.59 to 0.98. This means that one can be 95% confident that the true value of θ lies between 0.59 and 0.98. Suppose you did an orthodox analysis and found the same confidence interval. What is the orthodox interpretation of this interval?

Exercise 1.1.4 Suppose you learn that the questions are all true or false questions. Does this knowledge affect your prior distribution? And, if so, how would this prior in turn affect your posterior distribution?

1.2 Prediction

The posterior distribution θ contains all that we know about the rate with which you answer questions correctly. One way to use the knowledge is *prediction*.

For example, suppose you are confronted with a new set of 5 questions, all of the same difficulty as before. How can we formalize our expectations about your performance on this new set? In other words, how can we use the posterior distribution $p(\theta \mid n = 10, k = 9)$—which, after all, represents everything that we know about θ from the old set—to *predict* the number of correct responses out of the new set of $n^{\text{rep}} = 5$ questions? The mathematical solution is to integrate over the posterior,

$\int p\left(k^{\text{rep}} \mid \theta, n^{\text{rep}} = 5\right) p\left(\theta \mid n = 10, k = 9\right)\ \mathrm{d}\theta$, where k^{rep} is the predicted number of correct responses out of the additional set of 5 questions.

Computationally, you can think of this procedure as repeatedly drawing a random value θ_i from the posterior, and using that value to every time determine a single k^{rep}. The end result is $p\left(k^{\text{rep}}\right)$, the posterior predictive distribution of the possible number of correct responses in the additional set of 5 questions. The important point is that by integrating over the posterior, all predictive uncertainty is taken into account.

Exercise

Exercise 1.2.1 Instead of "integrating over the posterior," orthodox methods often use the "plug-in principle." In this case, the plug-in principle suggests that we predict $p(k^{\text{rep}})$ solely based on $\hat{\theta}$, the maximum likelihood estimate. Why is this generally a bad idea? Can you think of a specific situation in which this may not be so much of a problem?

1.3 Sequential updating

Bayesian analysis is particularly appropriate when you want to combine different sources of information. For example, assume that you are presented with a new set of 5 questions of equal difficulty. You answer 3 out of 5 correctly. How can we combine this new information with the old? Or, in other words, how do we update our knowledge of θ? Consistent with intuition, Bayes' rule entails that the prior that should be updated based on your performance for the new set is the posterior that was obtained based on your performance for the old set. Or, as Lindley put it, "today's posterior is tomorrow's prior" (Lindley, 1972, p. 2).

When all the data have been collected, however, the order in which this was done is irrelevant. The results from the 15 questions could have been analyzed as a single batch; they could have been analyzed sequentially, one-by-one; they could have been analyzed by first considering the set of 10 questions and next the set of 5, or vice versa. For all these cases, the end result, the final posterior distribution for θ, is identical. Given the same available information, Bayesian inference reaches the same conclusion, independent of the order in which the information was obtained. This again contrasts with orthodox inference, in which inference for sequential designs is radically different from that for non-sequential designs (for a discussion, see, for example, Anscombe, 1963).

Thus, a posterior distribution describes our uncertainty with respect to a parameter of interest, and the posterior is useful—or, as a Bayesian would have it, necessary—for probabilistic prediction and for sequential updating. To illustrate, in the case of our binomial example the uniform prior is a beta distribution with parameters $\alpha = 1$ and $\beta = 1$, and when combined with the binomial likelihood

this yields a posterior that is also a beta distribution, with parameters $\alpha + k$ and $\beta + n - k$. In simple *conjugate* cases such as these, where the prior and the posterior belong to the same distributional family, it is possible to obtain analytical solutions for the posterior distribution, but in many interesting cases it is not.

1.4 Markov chain Monte Carlo

In general, the posterior distribution, or any of its summary measures, can only be obtained analytically for a restricted set of relatively simple models. Thus, for a long time, researchers could only proceed easily with Bayesian inference when the posterior was available in closed-form or as a (possibly approximate) analytic expression. As a result, practitioners interested in models of realistic complexity did not much use Bayesian inference. This situation changed dramatically with the advent of computer-driven sampling methodology, generally known as Markov chain Monte Carlo (MCMC: e.g., Gamerman & Lopes, 2006; Gilks, Richardson, & Spiegelhalter, 1996). Using MCMC techniques such as Gibbs sampling or the Metropolis–Hastings algorithm, researchers can directly sample sequences of values from the posterior distribution of interest, forgoing the need for closed-form analytic solutions. The current adage is that *Bayesian models are limited only by the user's imagination*.

In order to visualize the increased popularity of Bayesian inference, Figure 1.2 plots the proportion of articles that feature the words "Bayes" or "Bayesian," according to Google Scholar (for a similar analysis for specific journals in statistics and economics see Poirier, 2006). The time line in Figure 1.2 also indicates the introduction of WinBUGS, a general-purpose program that greatly facilitates Bayesian analysis for a wide range of statistical models (Lunn, Thomas, Best, & Spiegelhalter, 2000; Lunn, Spiegelhalter, Thomas, & Best, 2009; Sheu & O'Curry, 1998). MCMC methods have transformed Bayesian inference to a vibrant and practical area of modern statistics.

For a concrete and simple illustration of Bayesian inference using MCMC, consider again the binomial example of 9 correct responses out of 10 questions, and the associated inference problem for θ, the rate of answering questions correctly. Throughout this book, we use WinBUGS to do Bayesian inference, saving us the effort of coding the MCMC algorithms ourselves.[1] Although WinBUGS does not work for every research problem application, it will work for many in cognitive sci-

[1] At this point, some readers want to know how exactly MCMC algorithms work. Other readers feel the urge to implement MCMC algorithms themselves. The details of MCMC sampling are covered in many other sources and we do not repeat that material here. We recommend the relevant chapters from the following books, listed in order of increasing complexity: Kruschke (2010a), MacKay (2003), Gilks et al. (1996), Ntzoufras (2009), and Gamerman and Lopes (2006). An introductory overview is given in Andrieu, De Freitas, Doucet, and Jordan (2003). You can also browse the internet, and find resources such as `http://www.youtube.com/watch?v=4gNpgSPal_8` and `http://www.learnbayes.org/`.

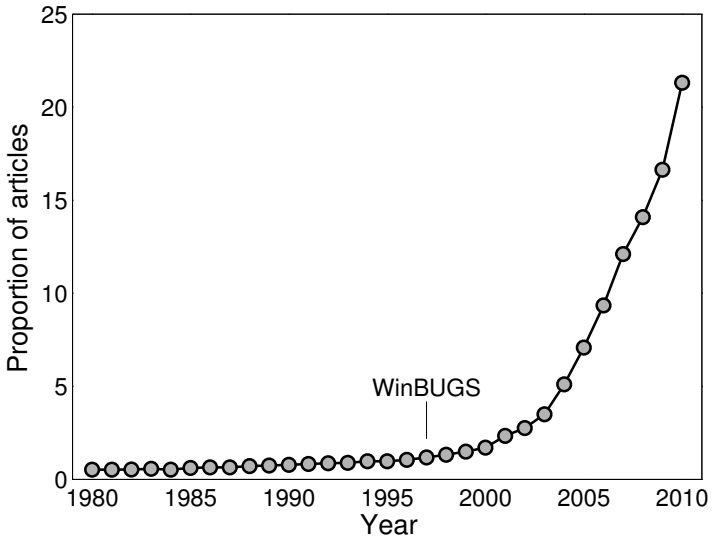


Fig. 1.2 A Google Scholar perspective on the increasing popularity of Bayesian inference, showing the proportion of articles matching the search "bayes OR bayesian -author: bayes" for the years 1980 to 2010.

ence. WinBUGS is easy to learn and is supported by a large community of active researchers.

The WinBUGS program requires you to construct a file that contains the model specification, a file that contains initial values for the model parameters, and a file that contains the data. The model specification file is most important. For our binomial example, we set out to obtain samples from the posterior of θ. The associated WinBUGS model specification code is two lines long:

```
model{
  theta ~ dunif(0,1) # the uniform prior for updating by the data
  k ~ dbin(theta,n)  # the data; in our example, k = 9 and n = 10
}
```

In this code, the "$\sim$" or twiddle symbol denotes "is distributed as", `dunif(a,b)` indicates the uniform distribution with parameters a and b, and `dbin(theta,n)` indicates the binomial distribution with rate θ and n observations. These and many other distributions are built in to the WinBUGS program. The "#" or hash sign is used for comments. As WinBUGS is a declarative language, the order of the two lines is inconsequential. Finally, note that the values for k and n are not provided in the model specification file. These values constitute the data and they are stored in a separate file.

When this code is executed, you obtain a sequence of MCMC samples from the posterior $p(\theta \mid D)$. Each individual sample depends only on the one that immediately preceded it, and this is why the entire sequence of samples is called a *chain*.

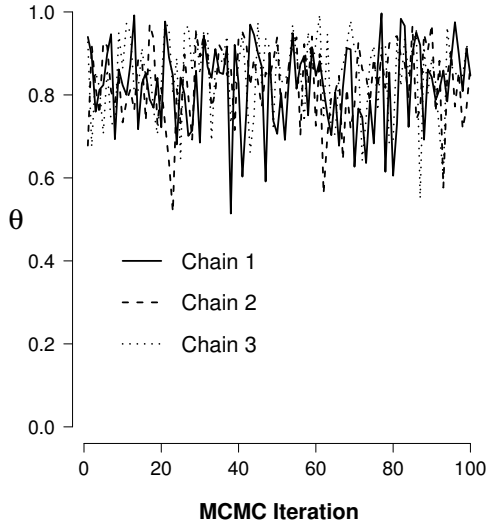


Fig. 1.3 Three MCMC chains for rate parameter θ, after observing 9 correct responses and 1 incorrect response.

In more complex models, it may take some time before a chain converges from its starting value to what is called its stationary distribution. To make sure that we only use those samples that come from the stationary distribution, and hence are unaffected by the starting values, it is good practice to diagnose convergence. This is an active area of research, and there is an extensive set of practical recommendations regarding achieving and measuring convergence (e.g., Gelman, 1996; Gelman & Hill, 2007).

A number of worked examples in this book deal with convergence issues in detail, but we mention three important concepts now. One approach is to run multiple chains, checking that their different initial starting values do not affect the distributions they sample from. Another is to discard the first samples from each chain, when those early samples are sensitive to the initial values. These discarded samples are called *burn-in* samples. Finally, it can also be helpful not to record every sample taken in a chain, but every second, or third, or tenth, or some other subset of samples. This is known as *thinning*, a procedure that is helpful when the chain moves slowly through the parameter space and, consequently, the current sample in the MCMC chain depends highly on the previous one. In such cases, the sampling process is said to be autocorrelated.

For example, Figure 1.3 shows the first 100 iterations for three chains that were set up to draw values from the posterior for θ. It is evident that the three chains are "mixing" well, suggesting early convergence. After assuring ourselves that the chains have converged, we can use the sampled values to plot a histogram, construct a density estimate, and compute values of interest. To illustrate, the three chains from Figure 1.3 were run for 3000 iterations each, for a total of 9000 samples from the posterior of θ. Figure 1.4 plots a histogram for the posterior. To visualize how the

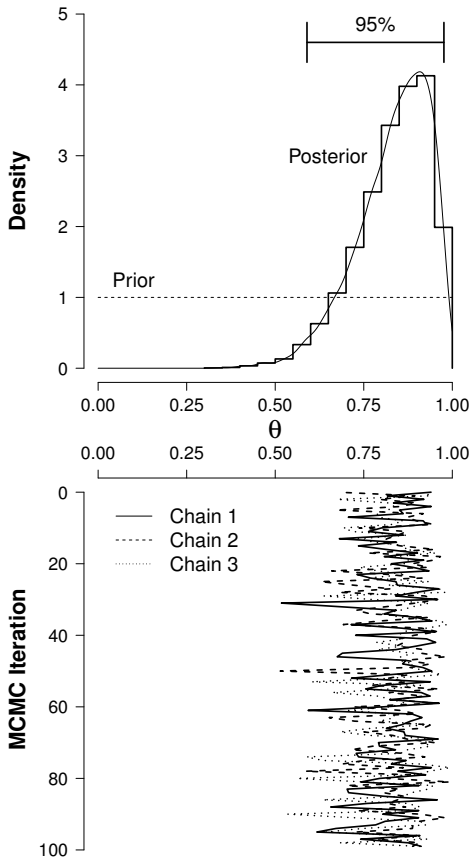


Fig. 1.4 MCMC-based Bayesian parameter estimation for rate parameter θ, after observing 9 correct responses and 1 incorrect response. The thin solid line indicates the fit of a density estimator. Based on this density estimator, the mode of the posterior distribution for θ is approximately 0.89, and the 95% credible interval extends from 0.59 to 0.98, closely matching the analytical results from Figure 1.1.

histogram is constructed from the MCMC chains, the bottom panel of Figure 1.4 plots the MCMC chains sideways; the histograms are created by collapsing the values along the "MCMC iteration" axis and onto the "θ" axis.

In the top panel of Figure 1.4, the thin solid line represent a density estimate. The mode of the density estimate for the posterior of θ is 0.89, whereas the 95% credible interval is $(0.59, 0.98)$, matching the analytical result shown in Figure 1.1.

The key point is that the analytical intractabilities that limited the scope of Bayesian parameter estimation have now been overcome. Using MCMC sampling, posterior distributions can be approximated to any desired degree of accuracy. This

Box 1.1 Why isn't every statistician a Bayesian?

"The answer is simply that statisticians do not know what the Bayesian paradigm says. Why should they? There are very few universities in the world with statistics departments that provide a good course in the subject. Only exceptional graduate students leave the field of their advisor and read for themselves. A secondary reason is that the subject is quite hard for someone who has been trained in the sampling-theory approach to understand. . . . The subject is difficult. Some argue that this is a reason for not using it. But it is always harder to adhere to a strict moral code than to indulge in loose living. . . . Every statistician would be a Bayesian if he took the trouble to read the literature thoroughly and was honest enough to admit that he might have been wrong." (Lindley, 1986, pp. 6–7).

book teaches you to use MCMC sampling and Bayesian inference to do research with cognitive science models and data.

Exercises

Exercise 1.4.1 Use Google and list some other scientific disciplines that use Bayesian inference and MCMC sampling.

Exercise 1.4.2 The text reads: "Using MCMC sampling, posterior distributions can be approximated to any desired degree of accuracy." How is this possible?

1.5 Goal of this book

The goal of this book is to show, by working through concrete examples, how Bayesian inference can be applied to modeling problems in cognitive science. Bayesian data analysis has received increasing attention from cognitive scientists, and for good reason.

1. Bayesian inference is *flexible*. This means that Bayesian models can respect the complexity of the data, and of the processes being modeled. For example, data analysis may require the inclusion of a contaminant process, a multi-level structure, or an account of missing data. Using the Bayesian approach, these sorts of additions are relatively straightforward.
2. Bayesian inference is *principled*. This means that all uncertainty is accounted for appropriately, and no useful information is discarded.

Box 1.2 Common sense expressed in numbers

"The Bayesian approach is a common sense approach. It is simply a set of techniques for orderly expression and revision of your opinions with due regard for internal consistency among their various aspects and for the data. Naturally, then, much that Bayesians say about inference from data has been said before by experienced, intuitive, sophisticated empirical scientists and statisticians. In fact, when a Bayesian procedure violates your intuition, reflection is likely to show the procedure to have been incorrectly applied." (Edwards et al., 1963, p. 195).

3. Bayesian inference yields *intuitive conclusions*. This reflects the fact that Bayesian inference is normative, stipulating how rational agents should change their opinion in the light of incoming data. Of course, it can nevertheless happen that you occasionally find a Bayesian conclusion to be surprising or counter-intuitive. You are then left with one of two options—either the analysis was not carried out properly (e.g., errors in coding, errors in model specification) or your intuition is in need of schooling.
4. Bayesian inference is *easy to undertake*. This means that with the software packages used in this book, Bayesian inference is often (but not always!) a trivial exercise. This frees up resources so more time can be spent on the substantive issues of developing theories and models, and interpreting results when they are applied to data.

At this point you may be champing at the bit, eager to apply the tools of Bayesian analysis to the kinds of cognitive models that interest you. But first we need to cover the basics and this is why Parts I, II, and III prepare you for the more complicated case studies presented in Part IV. This is not to say that the "elementary" material in Parts I, II, and III are devoid of cognitive context. On the contrary, we have tried to highlight how even the binomial model finds meaningful application in cognitive science.

Perhaps the material covered in this first chapter is still relatively abstract for you. Perhaps you are currently in a state of confusion. Perhaps you think that this book is too difficult, or perhaps you do not yet see clearly how Bayesian inference can help you in your own work. These feelings are entirely understandable, and this is why this book contains more than just this one chapter. Our teaching philosophy is that you learn the most by doing, not by reading. So if you still do not know exactly what a posterior distribution is, do not despair. The chapters in this book make you practice core Bayesian inference tasks so often that at the end you will know exactly what a posterior distribution is, whether you like it or not. Of course, we rather hope you like it, and we also hope that you will discover that Bayesian statistics can be exciting, rewarding, and, indeed, fun.

1.6 Further reading

This section provides some references for further reading. We first list Bayesian textbooks and seminal papers, then some texts that specifically deal with WinBUGS. We also note that Smithson (2010) presents a useful comparative review of six introductory textbooks on Bayesian methods.

1.6.1 Bayesian statistics

This section contains an annotated bibliography of Bayesian articles and books that we believe are particularly useful or inspiring.

- Berger, J. O. & Wolpert, R. L. (1988). *The Likelihood Principle* (2nd edn.). Hayward, CA: Institute of Mathematical Statistics. This is a great book if you want to understand the limitations of orthodox statistics. Insightful and fun.
- Bolstad, W. M. (2007). *Introduction to Bayesian Statistics* (2nd edn.). Hoboken, NJ: Wiley. Many books claim to introduce Bayesian statistics, but forget to state on the cover that the introduction is "for statisticians" or "for those comfortable with mathematical statistics." The Bolstad book is an exception, as it does not assume much background knowledge.
- Dienes, Z. (2008). *Understanding Psychology as a Science: An Introduction to Scientific and Statistical Inference*. New York: Palgrave Macmillan. An easy-to-understand introduction to inference that summarizes the differences between the various schools of statistics. No knowledge of mathematical statistics is required.
- Gamerman, D. & Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Boca Raton, FL: Chapman & Hall/CRC. This book discusses the details of MCMC sampling; a good book, but too advanced for beginners.
- Gelman, A. & Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. New York: Cambridge University Press. This book is an extensive practical guide on how to apply Bayesian regression models to data. WinBUGS code is provided throughout the book. Andrew Gelman also has an active blog that you might find interesting: `http://andrewgelman.com/`
- Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice*. Boca Raton, FL: Chapman & Hall/CRC. A citation classic in the MCMC literature, this book features many short chapters on all kinds of sampling-related topics: theory, convergence, model selection, mixture models, and so on.
- Gill, J. (2002). *Bayesian Methods: A Social and Behavioral Sciences Approach*. Boca Raton, FL: CRC Press. A well-written book that covers a lot of ground. Readers need some background in mathematical statistics.

- Hoff, P. D. (2009). *A First Course in Bayesian Statistical Methods*. Dordrecht, The Netherlands: Springer. A clear and well-written introduction to Bayesian inference, with accompanying R code, requiring some familiarity with mathematical statistics.
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge, UK: Cambridge University Press. Jaynes was one of the most ardent supporters of objective Bayesian statistics. The book is full of interesting ideas and compelling arguments, as well as being laced with Jaynes' acerbic wit, but it requires some mathematical background to appreciate all of the content.
- Jeffreys, H. (1939/1961). *Theory of Probability*. Oxford, UK: Oxford University Press. Sir Harold Jeffreys is the first statistician who exclusively used Bayesian methods for inference. Jeffreys also invented the Bayesian hypothesis test, and was generally far ahead of his time. The book is not always an easy read, in part because the notation is somewhat outdated. Strongly recommended, but only for those who already have a solid background in mathematical statistics and a firm grasp of Bayesian thinking. See `www.economics.soton.ac.uk/staff/aldrich/jeffreysweb.htm`
- Lee, P. M. (2012). *Bayesian Statistics: An introduction* (4th edn.). Chichester, UK: John Wiley. This well-written book illustrates the core tenets of Bayesian inference with simple examples, but requires a background in mathematical statistics.
- Lindley, D. V. (2000). The philosophy of statistics. *The Statistician, 49*, 293–337. One the godfathers of Bayesian statistics explains why Bayesian inference is right and everything else is wrong. Peter Armitage commented on the paper: "Lindley's concern is with the very nature of statistics, and his argument unfolds clearly, seamlessly and relentlessly. Those of us who cannot accompany him to the end of his journey must consider very carefully where we need to dismount; otherwise we shall find ourselves unwittingly at the bus terminus, without a return ticket."
- Marin, J.-M. & Robert, C. P. (2007). *Bayesian Core: A Practical Approach to Computational Bayesian Statistics*. New York: Springer. This is a good book by two reputable Bayesian statisticians. The book is beautifully typeset, includes an introduction to R, and covers a lot of ground. A firm knowledge of mathematical statistics is required. The exercises are challenging.
- McGrayne, S. B. (2011). *The Theory that Would not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy*. New Haven, CT: Yale University Press. A fascinating and accessible overview of the history of Bayesian inference.
- O'Hagan, A. & Forster, J. (2004). *Kendall's Advanced Theory of Statistics Vol. 2B: Bayesian Inference* (2nd edn.). London: Arnold. If you are willing to read only a single book on Bayesian statistics, this one is it. The book requires a background in mathematical statistics.
- Royall, R. M. (1997). *Statistical Evidence: A Likelihood Paradigm*. London: Chapman & Hall. This book describes the different statistical paradigms, and highlights

the deficiencies of the orthodox schools. The content can be appreciated without much background knowledge in statistics. The main disadvantage of this book is that the author is not a Bayesian. We still recommend the book, which is saying something.

1.6.2 WinBUGS texts

- Kruschke, J. K. (2010). *Doing Bayesian Data Analysis: A Tutorial Introduction with R and BUGS*. Burlington, MA: Academic Press. This is one of the first Bayesian books geared explicitly towards experimental psychologists and cognitive scientists. Kruschke explains core Bayesian concepts with concrete examples and OpenBUGS code. The book focuses on statistical models such as regression and ANOVA, and provides a Bayesian approach to data analysis in psychology, cognitive science, and empirical sciences more generally.
- Lee, S.-Y. (2007). *Structural Equation Modelling: A Bayesian Approach*. Chichester, UK: John Wiley. After reading the first few chapters from this book, you may wonder why not everybody uses WinBUGS for their structural equation modeling.
- Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2012). *The BUGS Book: A Practical Introduction to Bayesian Analysis*. Boca Raton, FL: Chapman & Hall/CRC Press. Quoted from the publisher: "Bayesian statistical methods have become widely used for data analysis and modelling in recent years, and the BUGS software has become the most popular software for Bayesian analysis worldwide. Authored by the team that originally developed this software, The BUGS Book provides a practical introduction to this program and its use. The text presents complete coverage of all the functionalities of BUGS, including prediction, missing data, model criticism, and prior sensitivity. It also features a large number of worked examples and a wide range of applications from various disciplines."
- Ntzoufras, I. (2009). *Bayesian Modeling using WinBUGS*. Hoboken, NJ: John Wiley. Provides an accessible introduction to WinBUGS. The book also presents a variety of Bayesian modeling examples, with an emphasis on Generalized Linear Models. See `www.ruudwetzels.com` for a detailed review.
- Spiegelhalter, D., Best, N., & Lunn, D. (2003). *WinBUGS User Manual 1.4*. Cambridge, UK: MRC Biostatistic Unit. Provides an introduction to WinBUGS, including a useful tutorial and various tips and tricks for new users. The user manual has effectively been superseded by *The BUGS Book* mentioned above.

2 Getting started with WinBUGS

with Dora Matzke

Throughout this book, you will use the WinBUGS (Lunn et al., 2000, 2009) software to work your way through the exercises. Although it is possible to do the exercises using the graphical user interface provided by the WinBUGS package, you can also use the Matlab or R programs to interact with WinBUGS.

In this chapter, we start by working through a concrete example using just WinBUGS. This provides an introduction to the WinBUGS interface, and the basic theoretical and practical components involved in Bayesian graphical model analysis. Completing the example will also quickly convince you that you do *not* want to rely on WinBUGS as your primary means for handling and analyzing data. It is not especially easy to use as a graphical user interface, and does not have all of the data management and visualization features needed for research.

Instead, we encourage you to choose either Matlab or R as your primary research computing environment, and use WinBUGS as an "add-on" that does the computational sampling part of analyses. Some WinBUGS interface capabilities will remain useful, especially in the exploratory stages of research. But either Matlab or R will be primary. Matlab and R code for every example in this book, as well as the scripts that implement the models in WinBUGS, are all available at `www.bayesmodels.com`.

This chapter first does a concrete example in WinBUGS, then re-works it in both Matlab and R. You should pay particular attention to the section that features your preferred research software. You will then be ready for the following chapters, which assume you are working in either Matlab or R, but understand the basics on the WinBUGS interface.

2.1 Installing WinBUGS, Matbugs, R, and R2WinBugs

2.1.1 Installing WinBUGS

WinBUGS is currently free software, and is available at `http://www.mrc-bsu.cam.ac.uk/bugs/`. Download the most recent version, including any patches, and make sure you download and apply the registration key. Some of the exercises in this book might work without the registration key, but some of them will not. You can download WinBUGS and the registration key directly from `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml`. A note to Windows 7 users: when you

install the patch and the registration key, make sure that you have first opened WinBUGS using the "Run as administrator" option (right-click on the WinBUGS icon to make this option available); next, go to `File` → `New`, copy-paste the code (i.e., patches or key), and then select `Tools` → `Decode` → `Decode All`.

2.1.2 Installing Matlab and Matbugs

Matlab is a commercial software package, and is available at `http://www.mathworks.com/`. As far as we know, any reasonably recent version of Matlab should let you do the exercises in this book. Also, as faras we know, no toolboxes are required. To give Matlab the ability to interact with WinBUGS, download the freely available `matbugs.m` function and put it in your Matlab working directory. You can download `matbugs.m` directly from `https://code.google.com/p/matbugs`.

2.1.3 Installing R and R2WinBUGS

R is a free software package, and is available at `http://www.r-project.org/`: click "download R," choose your download location, and proceed from there. Alternatively, you can download the Windows version of R directly from `http://cran.xl-mirror.nl/`. To give R the ability to interact with WinBUGS, you have to install the `R2WinBUGS package`. To install the `R2WinBUGS` package, start R and select the `Install Package(s)` option in the `Packages` menu. Once you choose your preferred CRAN mirror, select `R2WinBUGS` in the `Packages` window and click on `OK`.

2.2 Using the applications

2.2.1 An example with the binomial distribution

We will illustrate the use of WinBUGS, Matbugs, and R2WinBUGS by means of the same simple example from Chapter 1, which involved inferring the rate of success for a binary process. A binary process is anything where there are only two possible outcomes. An inference that is often important for these sorts of processes is the underlying rate at which the process takes one value rather than the other. Inferences about the rate can be made by observing how many times the process takes each value over a number of trials.

Suppose that one of the outcomes (e.g., the number of successes) happens on k out of n trials. These are known, or observed, data. The unknown variable of interest is the rate θ at which the outcomes are produced. Assuming that what happened on one trial does not influence the other trials, the number of successes k follows a binomial distribution, $k \sim \text{Binomial}(\theta, n)$. This relationship means that by observing the k successes out of n trials, it is possible to update our knowledge

Box 2.1 Our graphical model notation

There is no completely agreed standard notation for representing graphical models visually. It is always the case that nodes represent variables, and the graph structure connecting them represents dependencies. And it is almost always the case that plates are used to indicate replication. Beyond that core, there are regularities and family resemblances in the approaches used by numbers of authors and fields, but not adherence to a single standard. In this book, we make distinctions between: *continuous* versus *discrete* valued variables, using circular and square nodes; *observed* and *unobserved* variables, using shaded and unshaded nodes; and *stochastic* versus *deterministic* variables, using single- and double-bordered nodes. Alternative or additional conventions are possible, and could be useful. For example, our notation does not distinguish between observed variables that are data (e.g., the decision a subject makes in an experiment) and observed variables that are known properties of an experimental design (e.g., the number of trials a subject completes). It is also possible to argue that, for deterministic variables, it is the functions that are deterministic, and so the arrows in the graph, rather than the nodes, should be double-bordered.

about the rate θ. The basic idea of Bayesian analysis is that what we know, and what we do not know, about the variables of interest is always represented by probability distributions. Data like k and n allow us to update prior distributions for the unknown variables into posterior distributions that incorporate the new information.

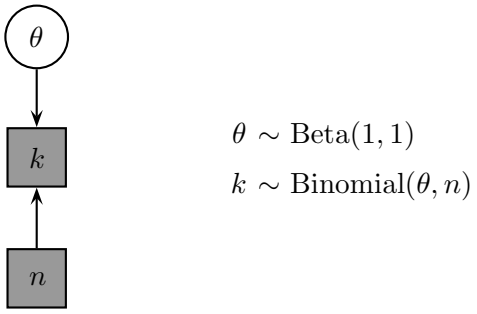


Fig. 2.1 Graphical model for inferring the rate of a binary process.

The graphical model representation of our binomial example is shown in Figure 2.1. The nodes represent all the variables that are relevant to the problem. The graph structure is used to indicate dependencies between the variables, with children depending on their parents. We use the conventions of representing unobserved

variables without shading and observed variables with shading, and continuous variables with circular nodes and discrete variables with square nodes.

Thus, the observed discrete numbers of successes k and number of trials n are represented by shaded and square nodes, and the unknown continuous rate θ is represented by an unshaded and circular node. Because the number of successes k depends on the number of trials n and on the rate of success θ, the nodes representing n and θ are directed towards the node representing k. We will start with the prior assumption that all possible rates between 0 and 1 are equally likely. We will thus assume a uniform prior $\theta \sim \text{Uniform}\big(0,1\big)$, which can equivalently be written in terms of a beta distribution as $\theta \sim \text{Beta}\big(1,1\big)$.

One advantage of using the language of graphical models is that it gives a complete and interpretable representation of a Bayesian probabilistic model. Another advantage is that WinBUGS can easily implement graphical models, and its various built-in MCMC algorithms are then able to do all of the inferences automatically.

2.2.2 Using WinBUGS

WinBUGS requires the user to construct three text files: one that contains the data, one that contains the starting values for the model parameters, and one that contains the model specification. The WinBUGS model code associated with our binomial example is available at `www.bayesmodels.com`, and is shown below:

```
# Inferring a Rate
model{
   # Prior Distribution for Rate Theta
   theta ~ dbeta(1,1)
   # Observed Counts
   k ~ dbin(theta,n)
}
```

Note that the uniform prior on θ is implemented here as $\theta \sim \text{Beta}\big(1,1\big)$. An alternative specification may seem more direct, namely $\theta \sim \text{Uniform}\big(0,1\big)$, denoted `dunif(0,1)` in WinBUGS. These two distributions are mathematically equivalent, but in our experience WinBUGS has fewer computational problems with the beta distribution implementation.

Implementing the model shown in Figure 2.1, and obtaining samples from the posterior distribution of θ, can be done by following the sequence of steps outlined below. At the present stage, do not worry about some of the finer details, as these will be clarified in the remainder of this book. Right now, the best you can do is simply to follow the instructions below and start clicking away.

1. Copy the model specification text above and paste it in a text file. Save the file, for example as `Rate_1.txt`.
2. Start WinBUGS. Open your newly created model specification file by selecting the `Open` option in the `File` menu, choosing the appropriate directory, and double-clicking on the model specification file. Do not forget to select files of type "txt," or you might be searching for a long time. Now check the syntax

of the model specification code by selecting the `Specification` option in the `Model` menu. Once the `Specification Tool` window is opened, as shown in Figure 2.2, highlight the word "model" at the beginning of the code and click on `check model`. If the model is syntactically correct and all parameters are given priors, the message "model is syntactically correct" will appear in the status bar all the way in the bottom left corner of the WinBUGS window. (But beware: the letters are very small and difficult to see.)

3. Create a text file that contains the data. The content of the file should look like this:

```
list(
k = 5,
n = 10
)
```

   Save the file, for example as `Data.Rate_1.txt`.

4. Open the data file and load the data. To open the data file, select the `Open` option in the `File` menu, select the appropriate directory, and double-click on the data file. To load the data, highlight the word "list" at the beginning of the data file and click on `load data` in the `Specification Tool` window, as shown in Figure 2.2. If the data are successfully loaded, the message "data loaded" will appear in the status bar.
5. Set the number of chains. Each chain is an independent run of the same model with the same data, although you can set different starting values for each chain.[1] Considering multiple chains provides a key test of convergence. In our binomial example, we will run two chains. To set the number of chains, type "2" in the field labelled `num of chains` in the `Specification Tool` window, shown in Figure 2.2.
6. Compile the model. To compile the model, click on `compile` in the `Specification Tool` window, shown in Figure 2.2. If the model is successfully compiled, the message "model compiled" will appear in the status bar.
7. Create a text file that contains the starting values of the unobserved variables (i.e., just the parameter θ for this model).[2] The content of the file should look like this:

```
list(
theta = 0.1
)
list(
theta = 0.9
)
```

[1] Running multiple chains is the best and easiest way to ensure WinBUGS uses different random number sequences in sampling. Doing a single-chain analysis multiple times can produce the same results because the random number sequence is identical.

[2] If you do not specify starting values yourself, WinBUGS will create them for you automatically. These automatic starting values are based on the prior and may occasionally result in numerical instability and program crashes. It is therefore safer to assign a starting value for all unobserved variables, and especially for variables at nodes "at the top" of the graphical model, which have no parents.

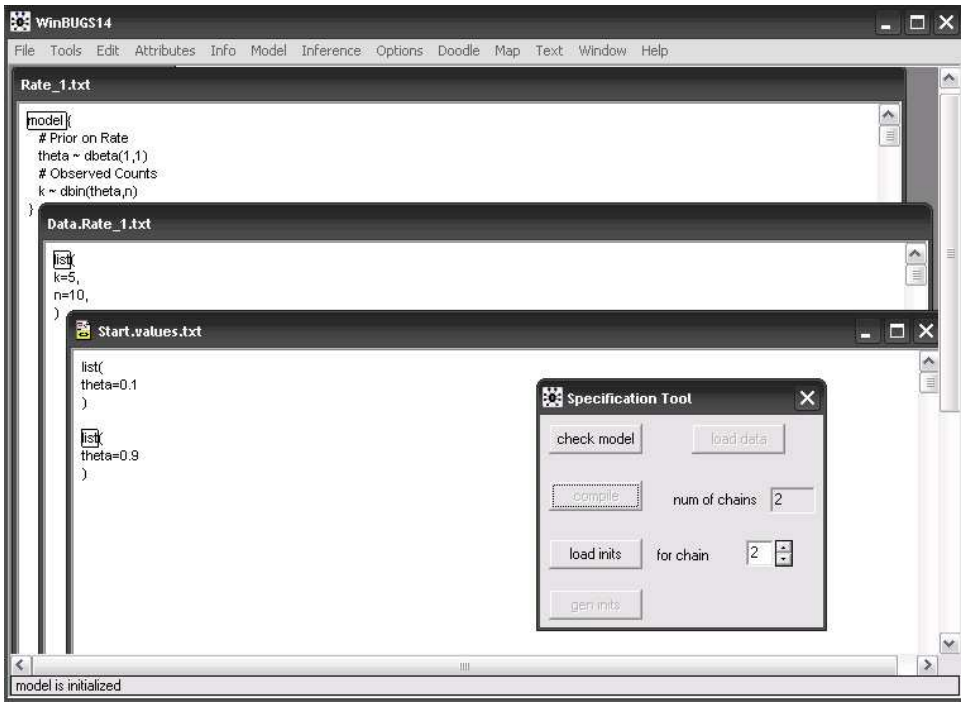


Fig. 2.2 The WinBUGS model specification tool.

Note that there are two initial values, one for each chain. Save the file, for example as `Start.values.txt`.

8. Open the file that contains the starting values by selecting the `Open` option in the `File` menu, selecting the appropriate directory, and double-clicking on the file. To load the starting value of θ for the first chain, highlight the word "list" at the beginning of the file and click on `load inits` in the `Specification Tool` window, shown in Figure 2.2. The status bar will now display the message "chain initialized but other chain(s) contain uninitialized variables." To load the starting value for the second chain, highlight the second "list" command and click on `load inits` once again. If all starting values are successfully loaded, the message "model is initialized" will appear in the status bar.
9. Set monitors to store the sampled values of the parameters of interest. To set a monitor for θ, select the `Samples` option from the `Inference` menu. Once the `Sample Monitor Tool` window, shown in Figure 2.3, is opened, type "theta" in the field labeled `node` and click on `set`.
10. Specify the number of samples you want to record. To do this, you first have to specify the total number of samples you want to draw from the posterior of θ, and the number of burn-in samples that you want to discard at the beginning of a sampling run. The number of recorded samples equals the total number of samples minus the number of burn-in samples. In our binomial example, we will

not discard any of the samples and will set out to obtain 20,000 samples from the posterior of θ. To specify the number of recorded samples, type "1" in the field labeled `beg` (i.e., WinBUGS will start recording from the first sample) and type "20000" in the field labeled `end` in the `Sample Monitor Tool` window, shown in Figure 2.3.

Fig. 2.3 The WinBUGS sample monitor tool.

11. Set "live" trace plots of the unobserved parameters of interest. WinBUGS allows you to monitor the sampling run in real-time. This can be useful on long sampling runs, for debugging, and for diagnosing whether the chains have converged. To set a "live" trace plot of θ, click on `trace` in the `Sample Monitor Tool` window, shown in Figure 2.3, and wait for an empty plot to appear on the screen. Once WinBUGS starts to sample from the posterior, the trace plot of θ will appear live on the screen.
12. Specify the total number of samples that you want to draw from the posterior. This is done by selecting the `Update` option from the `Model` menu. Once the `Update Tool` window, as in Figure 2.4, is opened, type "20000" in the field labeled `updates`. Typically, the number you enter in the `Update Tool` window will correspond to the number you entered in the `end` field of the `Sample Monitor Tool`.
13. Specify how many samples should be drawn between the recorded samples. You can, for example, specify that only every second drawn sample should be recorded. This ability to "thin" a chain is important when successive samples are not independent but autocorrelated. In our binomial example, we will record every sample that is drawn from the posterior of θ. To specify this, type "1" in the field labeled `thin` in the `Update Tool` window, shown in Figure 2.4, or in the `Sample Monitor Tool` window, shown in Figure 2.3. To record only every 10th sample, the `thin` field needs to be set to 10.
14. Specify the number of samples after which WinBUGS should refresh its display. To this end, type "100" in the field labeled `refresh` in the `Update Tool` window, shown in Figure 2.4.
15. Sample from the posterior. To sample from the posterior of θ, click on `update` in the `Update Tool` window, shown in Figure 2.4. During sampling, the message

"model is updating" will appear in the status bar. Once the sampling is finished, the message "updates took x s" will appear in the status bar.

Fig. 2.4 Update Tool.

16. Specify the output format. WinBUGS can produce two types of output; it can open a new window for each new piece of output, or it can paste all output into a single log file. To specify the output format for our binomial example, select `Output options` from the `Options` menu, and click on `log` in the `Output options` window.
17. Obtain summary statistics of the posterior distribution. To request summary statistics based on the sampled values of θ, select the `Samples` option in the `Inference` menu, and click on `stats` in the `Sample Monitor Tool` window, shown in Figure 2.3. WinBUGS will paste a table reporting various summary statistics for θ in the log file.
18. Plot the posterior distribution. To plot the posterior distribution of θ, click on `density` in the `Sample Monitor Tool` window, shown in Figure 2.3. WinBUGS will paste the "kernel density" of the posterior distribution of θ in the log file.[3]

Figure 2.5 shows the log file that contains the results for our binomial example. The first five lines of the log file document the steps taken to specify and initialize the model. The first output item is the `Dynamic trace` plot that allows the θ variable to be monitored during sampling, and is useful for diagnosing whether the chains have reached convergence. In this case, we can be reasonably confident that convergence has been achieved because the two chains, shown in different colors, are overlapping one another.[4] The second output item is the `Node statistics` table that presents the summary statistics for θ. Among other things, the table shows the mean, the standard deviation, and the median of the sampled values of θ. The last output item is the `Kernel density` plot that shows the posterior distribution of θ.

How did WinBUGS produce the results in Figure 2.5? The model specification file implemented the graphical model from Figure 2.1, saying that there is a rate θ with a uniform prior, that generates k successes out of n observations. The data file supplied the observed data, setting $k = 5$ and $n = 10$. WinBUGS then sampled

[3] A kernel density is a fancy smoothed histogram. Here, it is a smoothed histogram for the sampled values of θ.

[4] Note that the `Dynamic trace` plot only shows 200 samples. To have the entire time series of sampled values plotted in the log file, click on `history` in the `Sample Monitor Tool` window.

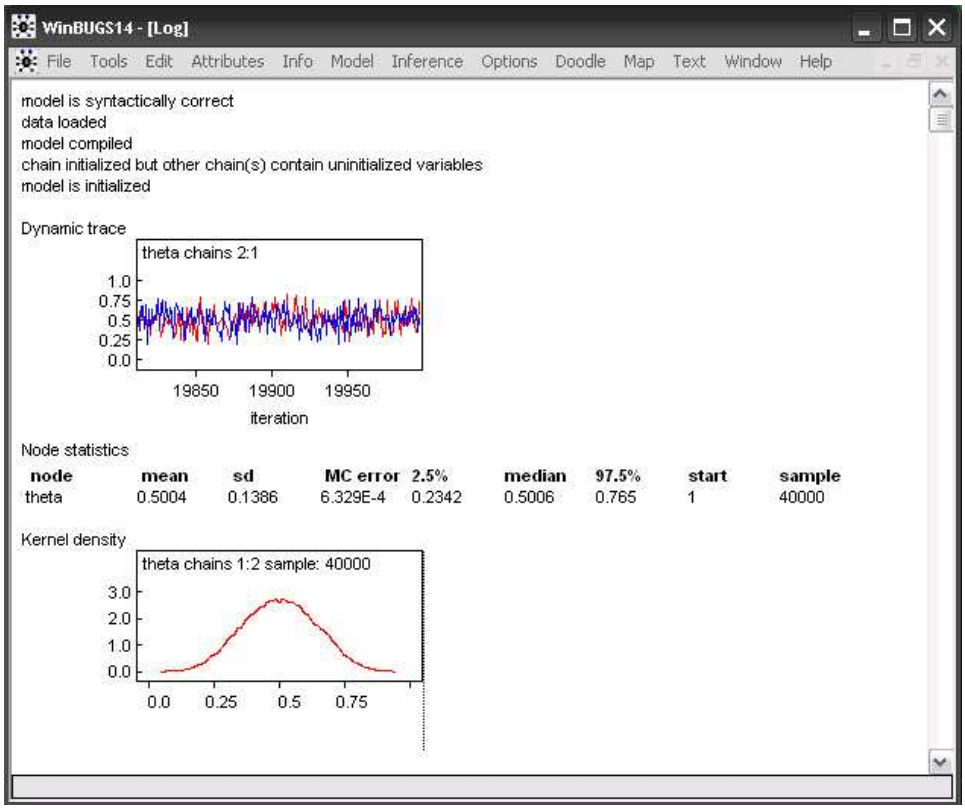


Fig. 2.5 Example of an output log file.

from the posterior of the unobserved variable θ. "Sampling" means drawing a set of values, so that the relative probability that any particular value will be sampled is proportional to the density of the posterior distribution at that value. For this example, the posterior samples for θ are a sequence of numbers like 0.5006, 0.7678, 0.3283, 0.3775, 0.4126, ... A histogram of these values is an approximation to the posterior distribution of θ.

Error messages

If the syntax of your model file is incorrect or the data and starting values are incompatible with your model specification, WinBUGS will balk and produce an error message. Error messages can provide useful information for debugging your WinBUGS code.[5] The error messages are displayed in the bottom left corner of the status bar, in very small letters.

Suppose, for example, that you mistakenly use the "assign" operator (`<-`) to specify the distribution of the prior on the rate parameter θ and the distribution of the observed data `k`:

[5] Although nobody ever accused WinBUGS of being user-friendly in this regard. Many error messages seem to have been written by the same people who did the Dead Sea Scrolls.

Box 2.2 Do I need or want to understand computational sampling?

Some people find the idea that WinBUGS looks after sampling, and that there is no need to understand the computational routines involved in detail, to be a relief. Others find it deeply disturbing. For the disturbed, there are many Bayesian texts that give detailed accounts of Bayesian inference using computational sampling. Start with the summary for cognitive scientists presented in Chapter 7 from Kruschke (2010a). Continue with the tutorial-style overview in Andrieu et al. (2003) or the relevant chapters in the excellent book by MacKay (2003), which is freely available on the Web, and move on to the more technical references such as Gilks et al. (1996), Ntzoufras (2009), and Gamerman and Lopes (2006). You can also browse the internet for more information; for example, there is an instructive applet at `http://www.lbreyer.com/classic.html`, and an excellent YouTube tutorial at `http://www.youtube.com/watch?v=4gNpgSPal_8`.

```
model{
  #Prior Distribution for Rate Theta
  theta <- dbeta(1,1)
  #Observed Counts
  k <- dbin(theta,n)
}
```

As WinBUGS requires you to use the tilde symbol "∼" to denote the distributions of the prior and the data, it will produce the following error message: `unknown type of logical function`, as shown in Figure 2.6. As another example, suppose that you mistype the distribution of the observed counts `k`, and you mistakenly specify the distribution of `k` as follows:

```
k ~ dbon(theta,n)
```

WinBUGS will not recognize `dbon` as an existing probability distribution, and will produce the following error message: `unknown type of probability density`, as shown in Figure 2.7.[6]

With respect to errors in the data file, suppose that your data file contains the following data: `k` = -5 and `n` = 10. Note, however, that `k` is the number of successes in the 10 trials and it is specified to be binomially distributed. WinBUGS therefore expects the value of `k` to lie between 0 and `n` and it will produce the following error message: `value of binomial k must be between zero and order of k`.

[6] On Windows machines, the error message is accompanied by a penetrating "system beep." After experiencing a few such system beeps you will want to turn them off. Browse the web for information on how to do this, or go straight to `http://www.howtogeek.com/howto/windows/turn-off-the-annoying-windows-xp-system-beeps/`. The people sitting next to you will be grateful too.

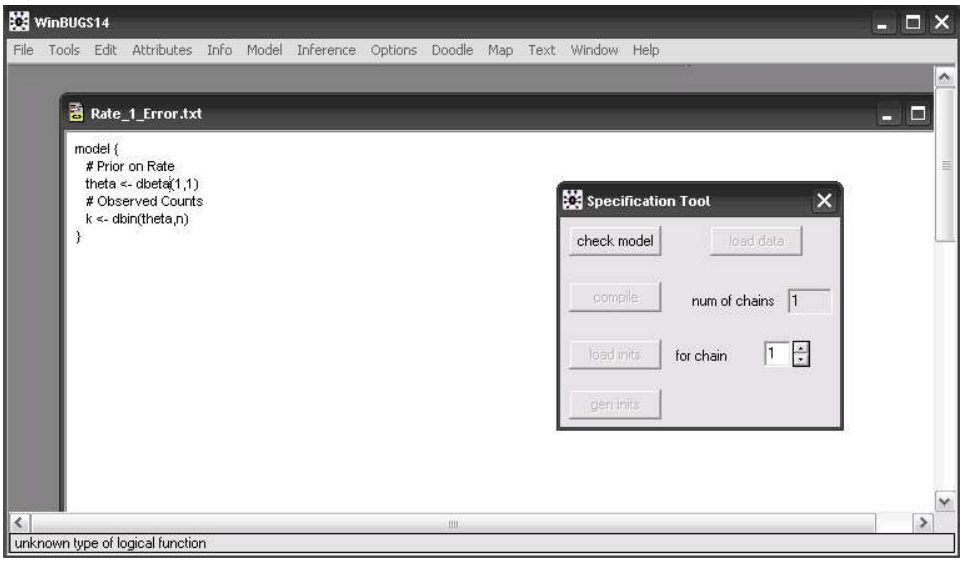


Fig. 2.6 WinBUGS error message as a result of incorrect logical operators. Note the small letters in the bottom left corner of the status bar.

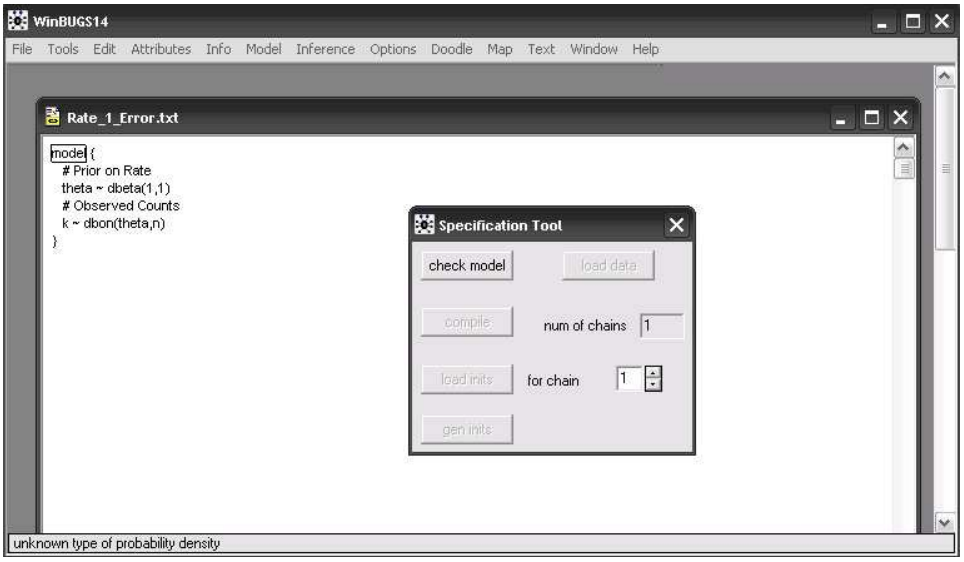


Fig. 2.7 WinBUGS error message as a result of a mis-specified probability density. Note the small letters in the bottom left corner of the status bar.

Finally, with respect to erroneous starting values, suppose that you chose 1.5 as the starting value of θ for the second chain. Because θ is the *probability* of getting 5 successes in 10 trials, WinBUGS expects the starting value for θ to lie between 0 and

Box 2.3 Changing the sampler

WinBUGS uses a suite of samplers, each fine-tuned to a particular class of statistical problems. Occasionally it may be worth the effort to change the default settings and edit the Updater/Rsrc/Methods.odc file. Any such editing should be done with care, and only after you have made a copy of the original Methods.odc file that contains the default settings. The advantage of changing the sampler is that it may circumvent traps or crashes. For example, the WinBUGS manual mentions that problems with the adaptive rejection sampler DFreeARS can sometimes be solved by replacing, for the log concave class, the method UpdaterDFreeARS by UpdaterSlice. Note for Windows 7 users: you may not be able to save any changes to files in the Updater/Rsrc directory. Work-around: copy the file to your desktop, edit it, save it, and copy it back to the Updater/Rsrc directory.

1. Therefore, specifying a value such as 1.5 produces the following error message: `value of proportion of binomial k must be between zero and one`.

2.2.3 Using Matbugs

We will use the `matbugs` function to call the WinBUGS software from within Matlab, and to return the results of the WinBUGS sampling to a Matlab variable for further analysis. The code we are using to do this is shown below:

```
% Data
k = 5;
n = 10;

% WinBUGS Parameters
nchains = 2; % How Many Chains?
nburnin = 0; % How Many Burn in Samples?
nsamples = 2e4;  % How Many Recorded Samples?
nthin = 1; % How Often is a Sample Recorded?

% Assign Matlab Variables to the Observed WinBUGS Nodes
datastruct = struct('k',k,'n',n);

% Initialize Unobserved Variables
start.theta= [0.1 0.9];

for i=1:nchains
    S.theta = start.theta(i); % An Intial Value for the Success Rate
    init0(i) = S;
end

% Use WinBUGS to Sample
[samples, stats] = matbugs(datastruct, ...
    fullfile(pwd, 'Rate_1.txt'),
```

```
'init', init0, 'view', 1, ...
'nChains', nchains, 'nburnin', nburnin, ...
'nsamples', nsamples, 'thin', nthin, ...
'DICstatus', 0, 'refreshrate',100, ...
'monitorParams', {'theta'}, ...
'Bugdir', 'C:/Program Files/WinBUGS14');
```

Some of the options in the Matbugs function control software input and output:

- `datastruct` contains the data that you want to pass from Matlab to WinBUGS.
- `fullfile` gives the name of the text file that contains the WinBUGS scripting of your graphical model (i.e., the model specification file).
- `view` controls the termination of WinBUGS. If `view` is set to 0, WinBUGS is closed automatically at the end of the sampling. If `view` is set to 1, WinBUGS remains open and it pastes the results of the sampling run in a log output file. To be able to inspect the results in WinBUGS, maximize the log output file and scroll up to the top of the page. Note that if you subsequently want WinBUGS to return the results to Matlab, you first have to close WinBUGS.
- `refreshrate` gives the number of samples after which WinBUGS should refresh its display.
- `monitorParams` gives the list of variables that will be monitored and returned to Matlab in the `samples` variable.
- `Bugdir` gives the location of the WinBUGS software.

Other options define the values for the computational sampling parameters:

- `init` gives the starting values for the unobserved variables.
- `nChains` gives the number of chains.
- `nburnin` gives the number of burn-in samples.
- `nsamples` gives the number of recorded samples that will be drawn from the posterior.
- `thin` gives the number of drawn samples between those that are recorded.
- `DICstatus` gives an option to calculate the Deviance Information Criterion (DIC) statistic (Spiegelhalter, Best, Carlin, & Van Der Linde, 2002). The DIC statistic is intended to be used for model selection, but is not universally accepted theoretically among Bayesian statisticians. If `DICstatus` is set to 0, the DIC statistic will not be calculated. If it is set to 1, WinBUGS will calculate the DIC statistic.

How did the WinBUGS script and Matlab work together to produce the posterior samples of θ? The WinBUGS model specification script defined the graphical model from Figure 2.1. The Matlab code supplied the observed data and the starting values for θ, and called WinBUGS. WinBUGS then sampled from the posterior of θ and returned the sampled values in the Matlab variable `samples.theta`. This flow of events is illustrated in Figure 2.9. You can plot the histogram of these sampled values using Matlab, in the way demonstrated in the script `Rate_1.m`. It should look something like the jagged line in Figure 2.8. Because the probability of any value

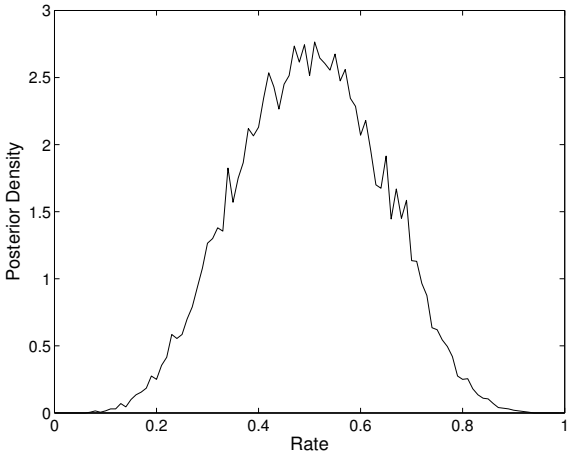


Fig. 2.8 Approximate posterior distribution of rate θ for $k = 5$ successes out of $n = 10$ trials, based on 20,000 posterior samples.

appearing in the sequence of posterior samples is decided by its relative posterior probability, the histogram is an approximation to the posterior distribution of θ.

Besides the sequence of posterior samples, WinBUGS also returns some useful summary statistics to Matlab. The variable `stats.mean` gives the mean of the posterior samples for each unobserved variable, which approximates its posterior expectation. This can often (but not always, as later exercises explore) be a useful point-estimate summary of all the information in the full posterior distribution. Similarly, `stats.std` gives the standard deviation of the posterior samples for each unobserved variable.

Finally, WinBUGS also returns the so-called $\hat{R}$ statistic in the `stats.Rhat` variable. This is a statistic about the sampling procedure itself, not about the posterior distribution. The $\hat{R}$ statistic is proposed by Brooks and Gelman (1998) and it gives information about convergence. The basic idea is to run two or more chains and measure the ratio of within-to-between-chain variance. If this ratio is close to 1, the independent sampling sequences are probably giving the same answer, and there is reason to trust the results.

Exercise

Exercise 2.2.1 Re-read the section on `view`. The Matlab code above specifies `view=1`. What does this do? Change the code to `view=0`. What has changed?

2.2.4 Using R2WinBUGS

We will use the `bugs()` function in the R2WinBUGS package to call the WinBUGS software from within R, and to return the results of the WinBUGS sampling to an

R variable for further analysis. Note for Windows 7 users: in order for the samples to be returned to R successfully, you may need to run R "as administrator" (right-click on the R icon to reveal this option). The R code we are using to obtain the WinBUGS samples is as follows:

```
setwd("D:/WinBUGS_Book/R_codes") #Set working directory, adjust as needed
library(R2WinBUGS) #Load the R2WinBUGS package
bugsdir <- "C:/Program Files/WinBUGS14" #Set WinBUGS directory, adjust as needed

k <- 5
n <- 10

data <- list("k", "n")
myinits <- list(
   list(theta = 0.1), #chain 1 starting value
   list(theta = 0.9)) #chain 2 starting value

parameters <- c("theta")

samples <- bugs(data, inits=myinits, parameters,
               model.file ="Rate_1.txt",
               n.chains=2, n.iter=20000, n.burnin=1, n.thin=1,
               DIC=T, bugs.directory=bugsdir,
               codaPkg=F, debug=F)
```

Note that lines 1 and 3 (i.e., the `setwd` line and the `bugsdir` line) specify the working directory and the WinBUGS directory, but only for the computer that runs the code. If you want to run the code on your own computer you need to modify these lines to match your setup.[7]

Some of the above options control software input and output:

- `data` contains the data that you want to pass from R to WinBUGS.
- `parameters` gives the list of variables that will be monitored and returned to R in the `samples` variable.
- `model.file` gives the name of the text file that contains the WinBUGS scripting of your graphical model (i.e., the model specification file). Avoid using non-alphanumeric characters (e.g., "&" and "*") in the directory and file names. Also, make sure that the name of the directory that contains the model file is not too long, otherwise WinBUGS will generate the following error message: `incompatible copy`. If WinBUGS fails to locate a correctly specified model file, try to include the entire path in the `model.file` argument.
- `bugs.directory` gives the location of the WinBUGS software.
- `codaPkg` controls the content of the variable that is returned from WinBUGS. If `codaPkg=F` (i.e., `codaPkg` is set to FALSE), WinBUGS returns a variable that contains the results of the sampling run. If `codaPkg=T` (i.e., `codaPkg` is set to TRUE), WinBUGS returns a variable that contains the file names of the

[7] When the code does not work immediately, check whether you have changed the directories correctly. The working directory should contain the model file, in this case `Rate_1.txt`, and the `bugsdir` variable should refer to the directory that contains the `WinBUGS14.exe` file.

WinBUGS outputs and the corresponding paths. You can access these output files by means of the R function `read.bugs()`.

- `debug` controls the termination of WinBUGS. If `debug` is set to FALSE, WinBUGS is closed automatically at the end of the sampling. If `debug` is set to TRUE, WinBUGS remains open and it pastes the results of the sampling run in a log output file. To be able to inspect the results in WinBUGS, maximize the log output file and scroll up to the top of the page. Note that if you subsequently want WinBUGS to return the results in the R `samples` variable, you first have to close WinBUGS. In general, you will not be able to use R again until after you terminate WinBUGS.

The other options define the values for the computational sampling parameters:

- `inits` assigns starting values to the unobserved variables. If you want WinBUGS to choose these starting values for you, replace `inits=myinits` in the call to bugs with `inits=NULL`.
- `n.chains` gives the number of chains.
- `n.iter` gives the number of samples that will be drawn from the posterior.
- `n.burnin` gives the number of burn-in samples.
- `n.thin` gives the number of drawn samples between those that are recorded.
- `DIC` gives an option to calculate the Deviance Information Criterion (DIC) statistic (Spiegelhalter et al., 2002). The DIC statistic is intended to be used for model selection, but is not universally accepted theoretically among Bayesian statisticians. If `DIC` is set to FALSE, the DIC statistic will not be calculated. If it is set to TRUE, WinBUGS will calculate the DIC statistic.[8]

WinBUGS returns the sampled values of θ in the R variable `samples`. You can access these values by typing `samples$sims.array` or `samples$sims.list`. The flow of events is illustrated in Figure 2.9.

You can use R to plot the histogram of sampled values of θ, as is demonstrated in the script `Rate_1.R`. In addition to the sequence of posterior samples, WinBUGS also returns to R some useful summary statistics. These summary statistics can be obtained by typing `samples` at the R prompt. When you run two or more chains, the `samples` command also provides the $\hat{R}$ statistic, introduced by Brooks and Gelman (1998). The $\hat{R}$ statistic provides information about the convergence of the sampling procedure, not about the posterior distribution. The basic idea is to run two or more chains and measure the ratio of within-to-between-chain variance. If this ratio is close to 1, the independent sampling sequences are probably giving the same answer, and there is reason to trust the results.

[8] For some reason, setting DIC equal to FALSE can lead to problems in the communication between R and WinBUGS. It is safest to set DIC equal to TRUE, even when you are not interested in the DIC.

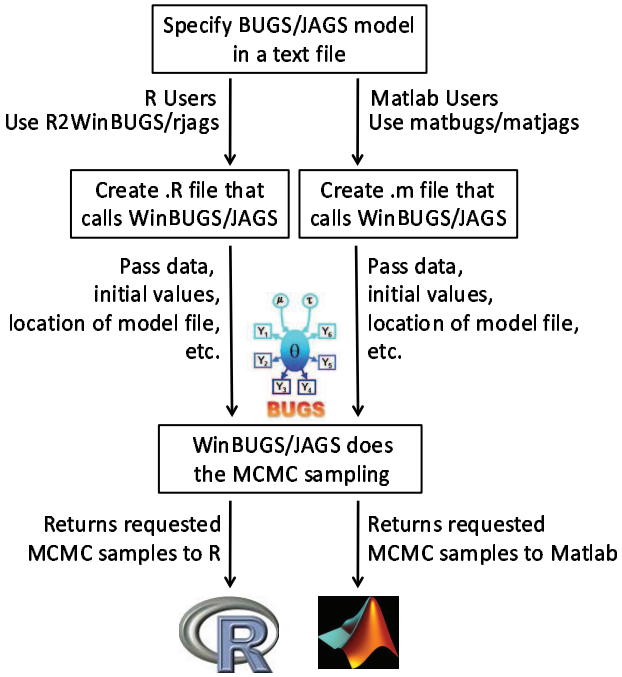


Fig. 2.9 Flowchart that illustrates the interaction between WinBUGS and R (left stream) or Matlab (right stream). JAGS is a program that is very similar to WinBUGS, described in the section on OpenBUGS and JAGS.

Exercise

Exercise 2.2.2 Re-read the section on `debug`. The R code above specifies `debug = F`; what does this do? Change the code to `debug = T`; what has changed?

2.3 Online help, other software, and useful URLs

2.3.1 Online help for WinBUGS

- The BUGS Project webpage `http://www.mrc-bsu.cam.ac.uk/bugs/weblinks/webresource.shtml` provides useful links to various articles, tutorial materials, and lecture notes about Bayesian modeling and the WinBUGS software.
- The BUGS discussion list `https://www.jiscmail.ac.uk/cgi-bin/webadmin?A0=bugs` is an online forum where WinBUGS users can exchange tips, ask questions, and share worked examples.

2.3.2 For Mac users

You can run WinBUGS on Macs using emulators, such as Darwine. As far as we know, you need a Dual Core Intel-based Mac and the latest stable version of Darwine to be able to use R2WinBUGS. Nevertheless, running WinBUGS on a Mac is not ideal. Mac users are encouraged to use JAGS instead. At the time of writing, the information below was useful for running WinBUGS on a Mac:

- The Darwine emulator is available at `www.kronenberg.org/darwine/`.
- The R2WinBUGS reference manual on the R-project webpage `cran.r-project.org/web/packages/R2WinBUGS/index.html` provides instructions on how to run R2winBUGS on Macs.
- Further information for running R2WinBUGS on Macs is available at `ggorjan.blogspot.com/2008/10/runnning-r2winbugs-on-mac.html` and `idiom.ucsd.edu/~rlevy/winbugsonmacosx.pdf`.
- Further information for running WinBUGS on Macs using a Matlab or R interface is available at `http://www.helensteingroever.com` and `www.ruudwetzels.com/macbugs`.

2.3.3 For Linux users

You can run WinBUGS under Linux using emulators, such as Wine and CrossOver.

- The BUGS Project webpage provides useful links to various examples on how to run WinBUGS under Linux `www.mrc-bsu.cam.ac.uk/bugs/faqs/contents.shtml` and how to run WinBUGS using a Matlab interface `www.mrc-bsu.cam.ac.uk/bugs/winbugs/remote14.shtml`.
- The R2WinBUGS reference manual on the R-project webpage `cran.r-project.org/web/packages/R2WinBUGS/index.html` provides instructions on how to run R2WinBUGS under Linux.

2.3.4 OpenBUGS, Stan, and JAGS

This book is designed primarily to work with WinBUGS. There are, however, alternative programs for generating MCMC samples from graphical models. Both OpenBUGS, Stan (Stan Development Team, 2013), and JAGS (Plummer, 2003) may be particularly attractive for Mac and Linux users, since they raise fewer issues than WinBUGS to install and run. The model code for OpenBUGS, Stan, and JAGS is very similar to WinBUGS, so that the transition from one program to the other is generally easy. An effort has been made to make most of the examples in this book compatible with JAGS. Often, in our experience, sampling is much faster in JAGS than it is in WinBUGS.

- OpenBUGS is available from `http://www.openbugs.info/w/`.
- Stan is available from `http://mc-stan.org/`.

- JAGS is available from `http://mcmc-jags.sourceforge.net/`.
- To give R the ability to interact with JAGS, you have to install the `rjags` package, and, optionally, the `R2jags` package. To ensure that you install the latest version of the `rjags` package, the safest procedure is to first Google the terms `rjags CRAN`, go to a website such as `http://cran.r-project.org/web/packages/rjags/index.html`, and—when using Windows—download the package zip file. Then start R, go to the `Packages` menu, choose `Install package(s) from local zip file...`, and select the package zip file you just downloaded. To check whether the installation was successful, type `library(rjags)` at the R prompt. To install the `R2jags` package, you can use the standard installation procedure: start R and select the `Install Package(s)` option in the `Packages` menu. After choosing your preferred CRAN mirror, select `R2jags` in the `Packages` window and click on `OK`.
- To give Matlab the ability to interact with JAGS, download the freely available `matjags.m` function and put it in your Matlab working directory. You can download `matjags.m` directly from `http://psiexp.ss.uci.edu/research/programs_data/jags/`.

PART II

PARAMETER ESTIMATION

Today's posterior is tomorrow's prior.

Lindley, 2000, p. 301

3 Inferences with binomials

3.1 Inferring a rate

Our first problem completes the introductory example in Chapter 2, and involves inferring the underlying success rate for a binary process. The graphical model is shown again in Figure 3.1. Recall that shaded nodes indicate known values, while unshaded nodes represent unknown values, and that circular nodes correspond to continuous values, while square nodes correspond to discrete values.

The goal of inference in the graphical model is to determine the posterior distribution of the rate θ, having observed k successes from n trials. The analysis starts with the prior assumption that all possible rates between 0 and 1 are equally likely. This corresponds to the uniform prior distribution $\theta \sim \text{Uniform}(0, 1)$, which can equivalently be written in terms of a beta distribution as $\theta \sim \text{Beta}(1, 1)$.

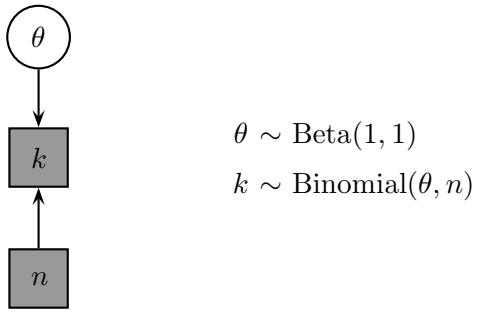


Fig. 3.1 Graphical model for inferring the rate θ of a binary process.

The script `Rate_1.txt` implements the graphical model in WinBUGS. The script is available at `www.bayesmodels.com` and is shown below:

```
# Inferring a Rate
model{
   # Prior Distribution for Rate Theta
   theta ~ dbeta(1,1)
   # Observed Counts
   k ~ dbin(theta,n)
}
```

The code `Rate_1.m` for Matlab or `Rate_1.R` for R, both available at `www.bayesmodels.com`, sets $k = 5$ and $n = 10$ and calls WinBUGS to sample from the graphical model. WinBUGS then returns to Matlab or R the posterior samples

Box 3.1 Beta distributions as conjugate priors

One of the nice properties of using the $\theta \sim \text{Beta}(\alpha, \beta)$ prior distribution for a rate θ is that it has a natural interpretation. The α and β values can be thought of as counts of, respectively, "prior successes" and "prior failures." This means that using a $\theta \sim \text{Beta}(3, 1)$ prior corresponds to having the prior information that 4 previous observations have been made, and 3 of them were successes. Or, more elaborately, starting with a $\theta \sim \text{Beta}(3, 1)$ is the same as starting with a $\theta \sim \text{Beta}(1, 1)$, and then seeing data giving two more successes (i.e., the posterior distribution in the second scenario will be the same as the prior distribution in the first). As always in Bayesian analysis, inference starts with prior information, and updates that information—by changing the probability distribution representing the uncertain information—as more information becomes available. When a type of likelihood function (in this case, the binomial) does not change the type of distribution (in this case, the beta) going from the prior to the posterior, they are said to have a "conjugate" relationship. This property is valued a lot in analytic approaches to Bayesian inference, because it makes for tractable calculations. It is not so important in the computational approaches emphasized in this book, because sampling methods can handle much more general relationships between parameter distributions and likelihood functions. But conjugacy is still useful in computational approaches because of the natural semantics it gives in setting prior distributions.

from θ. The Matlab or R code also plots the posterior distribution of the rate θ. A histogram of the samples looks something like the jagged line in Figure 3.2.

Exercises

Exercise 3.1.1 Carefully consider the posterior distribution for θ given $k = 5$ successes out of $n = 10$ trials. Based on a visual impression, what is your estimate of the probability that the rate θ is higher than 0.4 but smaller than 0.6? How did you arrive at your estimate?

Exercise 3.1.2 Consider again the posterior distribution for θ given $k = 5$ successes out of $n = 10$ trials. Based on a visual impression, what is your estimate of how much more likely it is that the rate θ is equal to 0.5 rather than 0.7? How did you arrive at your estimate?

Exercise 3.1.3 Alter the data to $k = 50$ and $n = 100$, and compare the posterior for the rate θ to the original with $k = 5$ and $n = 10$.

Exercise 3.1.4 For both the $k = 50$, $n = 100$ and $k = 5$, $n = 10$ cases just considered, re-run the analyses with many more samples (e.g., 10 times as many) by changing the `nsamples` variable in Matlab, or the `n.iter` variable

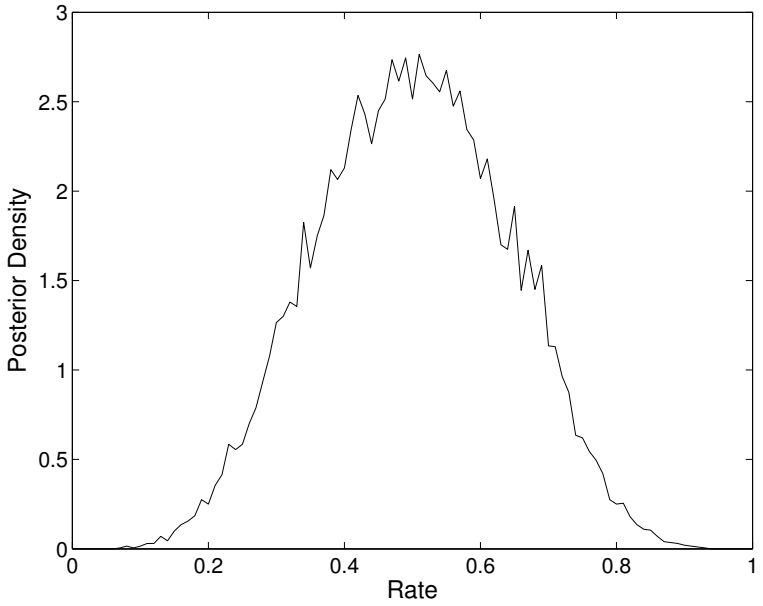


Fig. 3.2 Posterior distribution of rate θ for $k = 5$ successes out of $n = 10$ trials.

in R. This will take some time, but there is an important point to understand. What controls the width of the posterior distribution (i.e., the expression of uncertainty in the rate parameter θ)? What controls the quality of the approximation of the posterior (i.e., the smoothness of the histograms in the figures)?

Exercise 3.1.5 Alter the data to $k = 99$ and $n = 100$, and comment on the shape of the posterior for the rate θ.

Exercise 3.1.6 Alter the data to $k = 0$ and $n = 1$, and comment on what this demonstrates about the Bayesian approach.

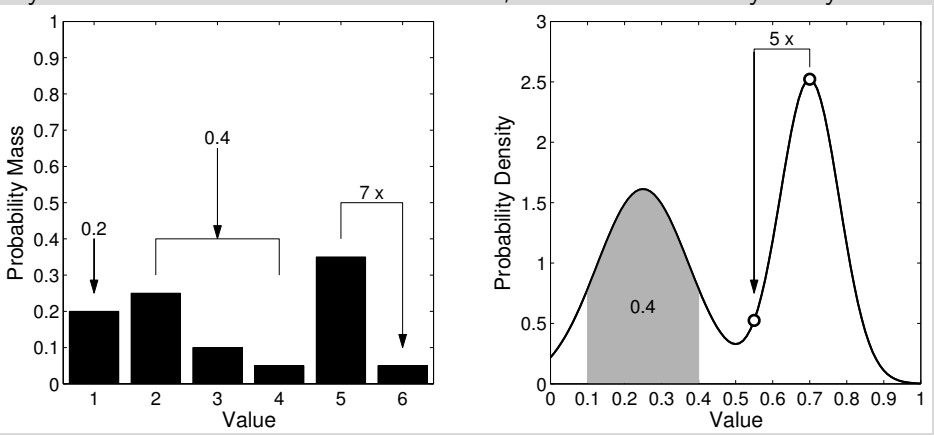
3.2 Difference between two rates

Now suppose that we have two different processes, producing k_1 and k_2 successes out of n_1 and n_2 trials, respectively. First, we will make the assumption that the underlying rates are different, so they correspond to different latent variables θ_1 and θ_2. Our interest is in the values of these rates, as estimated from the data, and in the difference $\delta = \theta_1 - \theta_2$ between the rates.

The graphical model representation for this problem is shown in Figure 3.3. The new notation is that the deterministic variable δ is shown by a double-bordered node. A deterministic variable is one that is defined in terms of other variables, and inherits its distribution from them. Computationally, deterministic nodes are

Box 3.2 Interpreting distributions

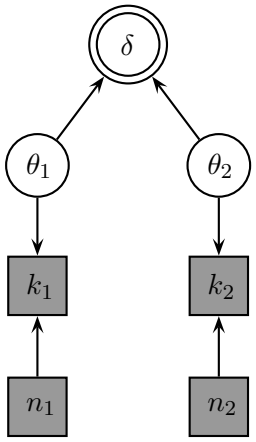
Since the essence of Bayesian inference is using probability distributions to represent uncertainty, it is important to be able to interpret probability mass functions and probability density functions. Probability mass functions are for discrete variables, which take a finite number of values, while probability density functions are for continuous variables, which take infinitely many values.



The panel on the left shows a probability mass function for a variable with 6 values. Each bar represents the probability of that value, so that, for example, the probability of the value 1 is 0.2. The probability of a range of values is the sum of their probabilities, so that the probability that the value is between 2 and 4 inclusive is 0.4. The ratio between the probabilities determines how much more likely one value is than another, so that the value 5 is 7 times more likely than the value 6. And, the sum of all of the probabilities (i.e., the height of the bars stacked on each other) is always 1. The panel on the right shows a probability density function for a variable that is between 0 and 1. The total area under the curve is always 1, which means the densities of individual points can (and often do) exceed 1. They cannot be interpreted as probabilities. But the probability of a range of values can be determined by the relevant area under the curve. In the right panel, the probability that the value is between 0.1 and 0.4 is 0.4. And ratios can still be interpreted in a relative way, so it is 5 times more likely the value is 0.7 than 0.55.

unnecessary—all inference could be done with the variables that define them—but they are often conceptually very useful to include, to communicate the meaning of a model.

The script `Rate_2.txt` implements the graphical model in WinBUGS:



$$
\begin{aligned}
k_1 &\sim \text{Binomial}(\theta_1, n_1) \\
k_2 &\sim \text{Binomial}(\theta_2, n_2) \\
\theta_1 &\sim \text{Beta}(1, 1) \\
\theta_2 &\sim \text{Beta}(1, 1) \\
\delta &\leftarrow \theta_1 - \theta_2
\end{aligned}
$$

Fig. 3.3 Graphical model for inferring the difference, $\delta = \theta_1 - \theta_2$, in the rates of two binary processes.

```
# Difference Between Two Rates
model{
   # Observed Counts
   k1 ~ dbin(theta1,n1)
   k2 ~ dbin(theta2,n2)
   # Prior on Rates
   theta1 ~ dbeta(1,1)
   theta2 ~ dbeta(1,1)
   # Difference Between Rates
   delta <- theta1-theta2
}
```

The code `Rate_2.m` or `Rate_2.R` sets $k_1 = 5$, $k_2 = 7$, $n_1 = n_2 = 10$, and then calls WinBUGS to sample from the graphical model. WinBUGS returns to Matlab or R the posterior samples from θ_1, θ_2, and δ. If the main research question is how different the rates are, then δ is the most relevant variable, and its posterior distribution is shown in Figure 3.4.

There are many ways the full information in the posterior distribution of δ might usefully be summarized. The Matlab or R code produces a set of these from the posterior samples, including:

- The mean value, which approximates the expectation of the posterior. This summary tries to pick a single value close to the truth, with bigger deviations from the truth being punished more heavily. Statistically, it corresponds to the point estimate under quadratic loss.
- The value with maximum density in the posterior samples, approximating the posterior mode. This summary aims to pick the single most likely value. This is known as the maximum a posteriori (MAP) estimate, and is the same as the maximum likelihood estimate (MLE) for "flat" priors. Statistically, it corresponds to the point estimate under zero–one loss. Estimating the mode requires

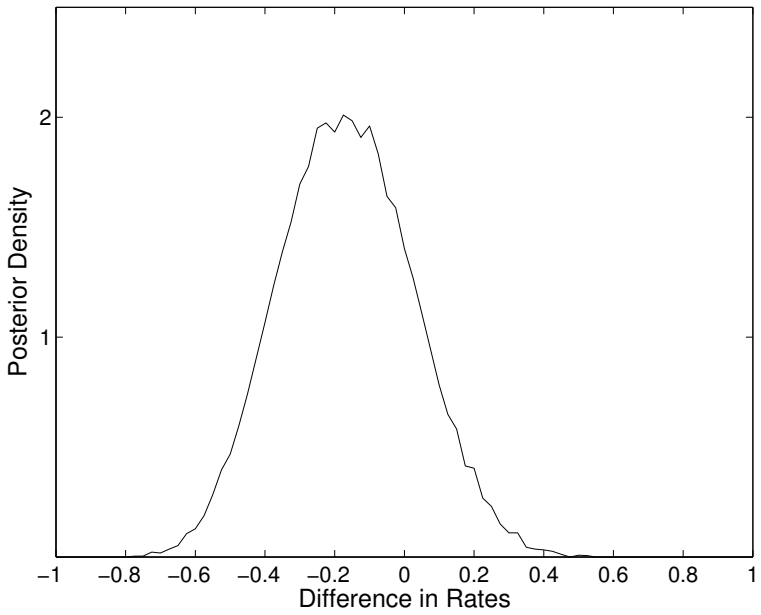


Fig. 3.4 Posterior distribution of the difference between two rates $\delta = \theta_1 - \theta_2$.

evaluating the likelihood function at each posterior sample, and so requires a bit more post-processing work in Matlab or R.

- The median value, which is the value that separates the highest 50% of the posterior distribution from the lowest 50%, and so finds the middle-most value. Statistically, it corresponds to the point estimate under linear loss.
- The 95% credible interval. This gives the upper and lower values between which 95% of samples fall. Thus, it approximates the bounds on the posterior distribution that contain 95% of the posterior density. The Matlab or R code can be modified to produce credible intervals for criteria other than 95%.

For the current problem, the mean of δ estimated from the returned samples is approximately -0.17, the mode is approximately -0.17, the median is approximately -0.17, and the 95% credible interval is approximately $[-0.52, 0.21]$.

Exercises

Exercise 3.2.1 Compare the data sets $k_1 = 8$, $n_1 = 10$, $k_2 = 7$, $n_2 = 10$ and $k_1 = 80$, $n_1 = 100$, $k_2 = 70$, $n_2 = 100$. Before you run the code, try to predict the effect that adding more trials has on the posterior distribution for δ.

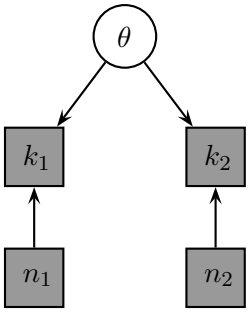
Exercise 3.2.2 Try the data $k_1 = 0$, $n_1 = 1$ and $k_2 = 0$, $n_2 = 5$. Can you explain the shape of the posterior for δ?

Exercise 3.2.3 In what context might different possible summaries of the posterior distribution of δ (i.e., point estimates, or credible intervals) be reasonable, and when might it be important to show the full posterior distribution?

3.3 Inferring a common rate

We continue to consider two binary processes, producing k_1 and k_2 successes out of n_1 and n_2 trials, respectively, but now assume the underlying rate for both is the same. This means there is just one rate, θ.

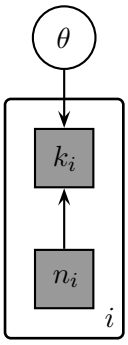
The graphical model representation for this problem is shown in Figure 3.5.



$$k_1 \sim \text{Binomial}(\theta, n_1)$$
$$k_2 \sim \text{Binomial}(\theta, n_2)$$
$$\theta \sim \text{Beta}(1, 1)$$

Fig. 3.5 Graphical model for inferring the common rate θ of two binary processes.

An equivalent graphical model, using plate notation, is shown in Figure 3.6. Plates are bounding rectangles that enclose independent replications of a graphical structure within a whole model. In this case, the plate encloses the two observed counts and numbers of trials. Because there is only one latent rate θ (i.e., the same probability drives both binary processes) it is not iterated inside the plate. One way to think of plates, which some people find helpful, is as "for loops" from programming languages (including WinBUGS itself).



$$k_i \sim \text{Binomial}(\theta, n_i)$$
$$\theta \sim \text{Beta}(1, 1)$$

Fig. 3.6 Graphical model for inferring the common rate θ underlying a number of binary processes, using plate notation.

The script `Rate_3.txt` implements the graphical model in WinBUGS:

```
# Inferring a Common Rate
model{
   # Observed Counts
```

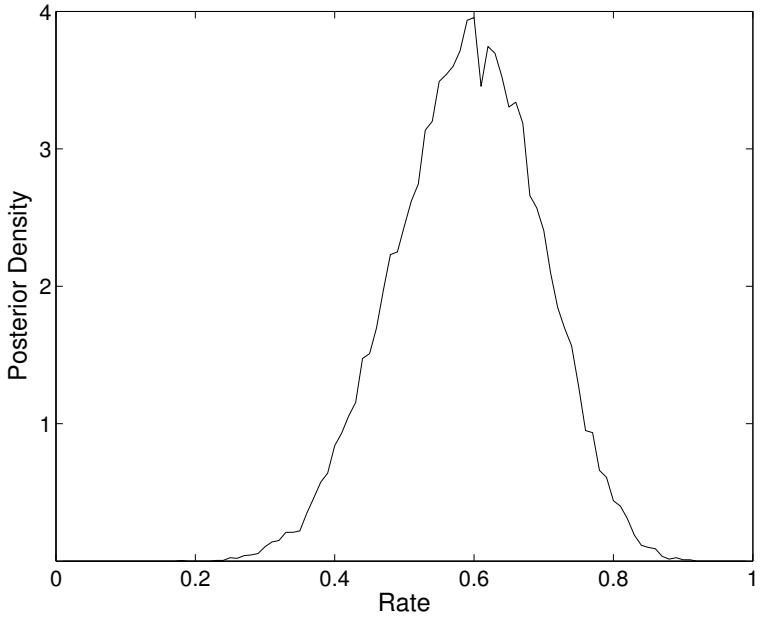


Fig. 3.7 Posterior distribution of the common rate θ of two binary processes.

```
   k1 ~ dbin(theta,n1)
   k2 ~ dbin(theta,n2)
   # Prior on Single Rate Theta
   theta ~ dbeta(1,1)
}
```

The code `Rate_3.m` or `Rate_3.R` sets k_1, k_2, n_1, and n_2, and then calls WinBUGS to sample from the graphical model.[1] The code also produces a plot of the posterior distribution for the common rate, as shown in Figure 3.7.

Exercises

Exercise 3.3.1 Try the data $k_1 = 14$, $n_1 = 20$, $k_2 = 16$, $n_2 = 20$. How could you report the inference about the common rate θ?

Exercise 3.3.2 Try the data $k_1 = 0$, $n_1 = 10$, $k_2 = 10$, $n_2 = 10$. What does this analysis infer the common rate θ to be? Do you believe the inference?

Exercise 3.3.3 Compare the data sets $k_1 = 7$, $n_1 = 10$, $k_2 = 3$, $n_2 = 10$ and $k_1 = 5$, $n_1 = 10$, $k_2 = 5$, $n_2 = 10$. Make sure, following on from the previous question, that you understand why the comparison works the way it does.

[1] Note that the R code specifies `debug=T`, and this means that WinBUGS needs to be closed (not minimized) before the sampling information can be returned to R. WinBUGS is ready as soon as the message "updates took x s" appears in the status bar.

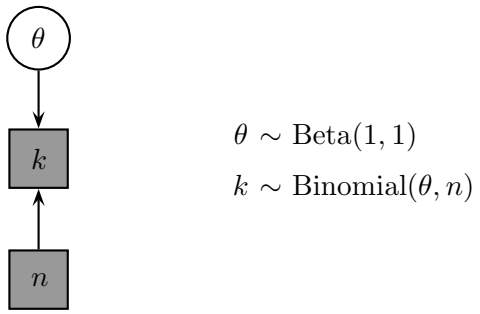


Fig. 3.8 Graphical model for inferring the rate θ of a binary process.

3.4 Prior and posterior prediction

One conceptual way to think about Bayesian analysis is that Bayes' rule provides a bridge between the unobserved parameters of models and the observed data. The most useful part of this bridge is that data allow us to update the uncertainty, represented by probability distributions, about parameters. But the bridge can handle two-way traffic, and so there is a richer set of possibilities for relating parameters to data. There are really four distributions available, and they are all important and useful.

- First, the *prior distribution* over parameters captures our initial assumptions or state of knowledge about the psychological variables they represent.
- Secondly, the *prior predictive distribution* tells us what data to expect, given our model and our current state of knowledge. The prior predictive is a distribution over data, and gives the relative probability of different observable outcomes before any data have been seen.
- Thirdly, the *posterior distribution* over parameters captures what we know about the psychological variables having updated the prior information with the evidence provided by data.
- Finally, the *posterior predictive distribution* tells us what data to expect, given the same model we started with, but with a current state of knowledge that has been updated by the observed data. Again, the posterior predictive is a distribution over data, and gives the relative probability of different observable outcomes after data have been seen.

As an example to illustrate these distributions, we return to the simple problem of inferring a single underlying rate. Figure 3.8 presents the graphical model, and is the same as Figure 3.1.

The script `Rate_4.txt` implements the graphical model in WinBUGS, and provides sampling not just for the posterior, but also for the prior, prior predictive, and posterior predictive:

```
# Prior and Posterior Prediction
model{
   # Observed Data
   k ~ dbin(theta,n)
   # Prior on Rate Theta
   theta ~ dbeta(1,1)
   # Posterior Predictive
   postpredk ~ dbin(theta,n)
   # Prior Predictive
   thetaprior ~ dbeta(1,1)
   priorpredk ~ dbin(thetaprior,n)
}
```

Posterior predictive sampling is achieved by the variable `postpredk` that samples predicted data using the same binomial as the actual observed data. To allow sampling from the prior, we use a dummy variable `thetaprior` that is identical to the one we actually do inference on, but is itself independent of the data, and so is never updated. Prior predictive sampling is achieved by the variable `priorpredk` that samples data using the same binomial, but relying on the prior rate.

The code `Rate_4.m` or `Rate_4.R` sets observed data with $k = 1$ successes out of $n = 15$ observations, and then calls WinBUGS to sample from the graphical model. The code also draws the four distributions, two in the parameter space (the prior and posterior for θ), and two in the data space (the prior predictive and posterior predictive for k). It should look something like Figure 3.9.

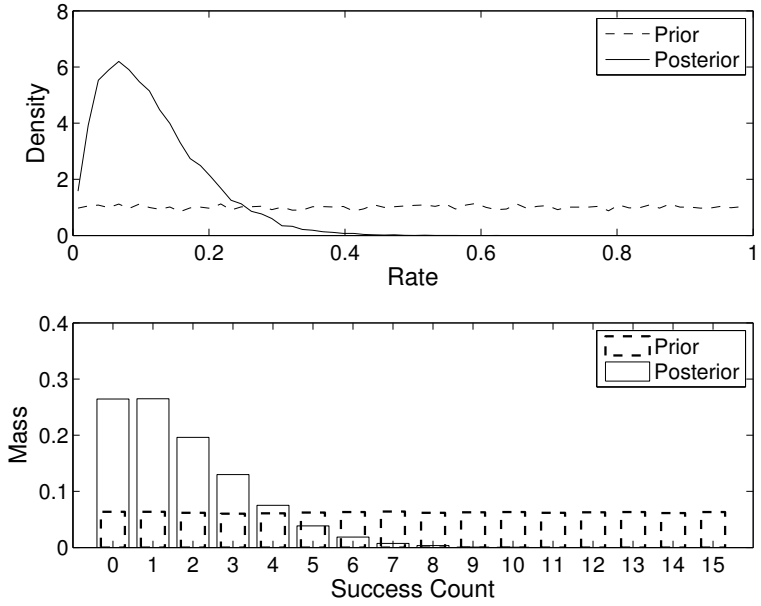


Fig. 3.9 Prior and posterior for the success rate θ (top panel), and prior and posterior predictive for counts of the number of successes (bottom panel), based on data giving $k = 1$ successes out of $n = 15$ trials.

Exercises

Exercise 3.4.1 Make sure you understand the prior, posterior, prior predictive, and posterior predictive distributions, and how they relate to each other (e.g., why is the top panel of Figure 3.9 a line plot, while the bottom panel is a bar graph?). Understanding these ideas is a key to understanding Bayesian analysis. Check your understanding by trying other data sets, varying both k and n.

Exercise 3.4.2 Try different priors on θ, by changing $\theta \sim \text{Beta}\big(1, 1\big)$ to $\theta \sim \text{Beta}\big(10, 10\big)$, $\theta \sim \text{Beta}\big(1, 5\big)$, and $\theta \sim \text{Beta}\big(0.1, 0.1\big)$. Use the figures produced to understand the assumptions these priors capture, and how they interact with the same data to produce posterior inferences and predictions.

Exercise 3.4.3 Predictive distributions are not restricted to exactly the same experiment as the observed data, and can be used in the context of any experiment where the inferred model parameters make predictions. In the current simple binomial setting, for example, predictive distributions could be found by an experiment that is different because it has $n' \neq n$ observations. Change the graphical model, and Matlab or R code, to implement this more general case.

Exercise 3.4.4 In October 2009, the Dutch newspaper *Trouw* reported on research conducted by H. Trompetter, a student from the Radboud University in the city of Nijmegen. For her undergraduate thesis, Trompetter had interviewed 121 older adults living in nursing homes. Out of these 121 older adults, 24 (about 20%) indicated that they had at some point been bullied by their fellow residents. Trompetter rejected the suggestion that her study may have been too small to draw reliable conclusions: "If I had talked to more people, the result would have changed by one or two percent at the most." Is Trompetter correct? Use the code `Rate_4.m` or `Rate_4.R`, by changing the `dataset` variable (Matlab) or changing the values for `k` and `n` (R), to find the prior and posterior predictive for the relevant rate parameter and bullying counts. Based on these distributions, do you agree with Trompetter's claims?

3.5 Posterior prediction

One important use of posterior predictive distributions is to examine the descriptive adequacy of a model. It can be viewed as a set of predictions about what data the model expects to see, based on the posterior distribution over parameters. If these predictions do not match the data already seen, the model is descriptively inadequate.

As an example to illustrate this idea of checking model adequacy, we return to the problem of inferring a common rate underlying two binary processes. Figure 3.10 presents the graphical model, and is the same as Figure 3.5.

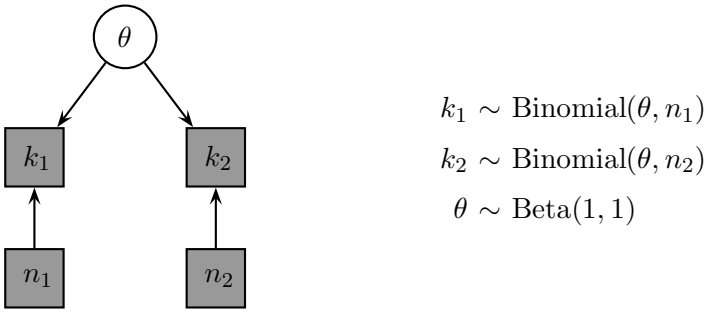


Fig. 3.10 Graphical model for inferring the common rate θ underlying two binary processes.

The script `Rate_5.txt` implements the graphical model in WinBUGS, and provides sampling for the posterior predictive distribution:

```
# Inferring a Common Rate, With Posterior Predictive
model{
   # Observed Counts
   k1 ~ dbin(theta,n1)
   k2 ~ dbin(theta,n2)
   # Prior on Single Rate Theta
   theta ~ dbeta(1,1)
   # Posterior Predictive
   postpredk1 ~ dbin(theta,n1)
   postpredk2 ~ dbin(theta,n2)
}
```

The code `Rate_5.m` or `Rate_5.R` sets observed data with $k_1 = 0$ successes out of $n_1 = 10$ observations, and $k_2 = 10$ successes out of $n_2 = 10$ observations, as considered in Exercise 3.3.2. The code draws the posterior distribution for the rate and the posterior predictive distribution, as shown in Figure 3.11.

The left panel shows the posterior distribution over the common rate θ for two binary processes, which gives density to values near 0.5. The right panel shows the posterior predictive distribution of the model, with respect to the two success counts. The size of each square is proportional to the predictive mass given to each

Box 3.3 The fundamental problem of inference

"The fundamental problem of inference and induction is to use past data to predict future data. Extensive observations on the motions of heavenly bodies enables their future positions to be calculated. Clinical studies on a drug allow a doctor to give a prognosis for a patient for whom the drug is prescribed. Sometimes the uncertain data are in the past, not the future. A historian will use what evidence he has to assess what might have happened where records are missing. A court of criminal law enquires about what had happened on the basis of later evidence." (Lindley, 2000, p. 304).

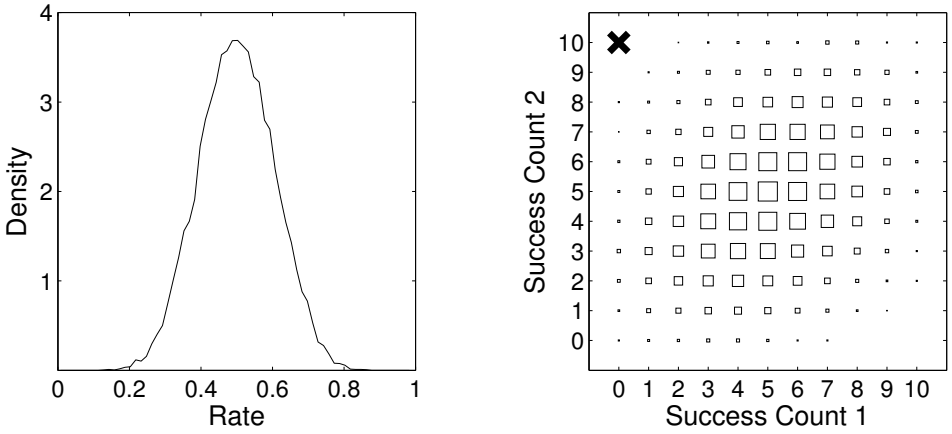


Fig. 3.11 The posterior distribution of the common rate θ for two binary processes (left panel), and the posterior predictive distribution (right panel), based on 0 and 10 successes out of 10 observations.

possible combination of success count observations. The actual data observed in this example, with 0 and 10 successes for the two counts, are shown by the cross.

Exercises

Exercise 3.5.1 Why is the posterior distribution in the left panel inherently one-dimensional, but the posterior predictive distribution in the right panel inherently two-dimensional?

Exercise 3.5.2 What do you conclude about the descriptive adequacy of the model, based on the relationship between the observed data and the posterior predictive distribution?

Exercise 3.5.3 What can you conclude about the parameter θ?

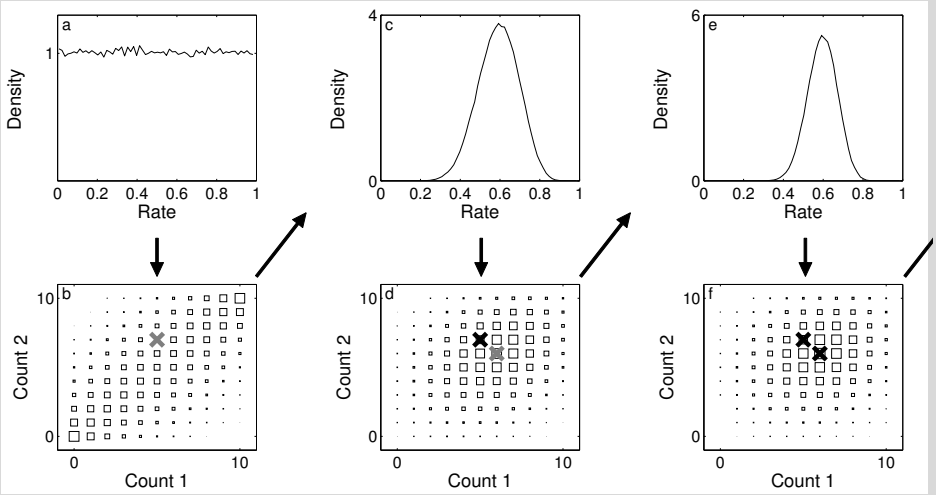
3.6 Joint distributions

So far, we have assumed that the number of successes k and number of total observations n is known, but that the underlying rate θ is unknown. This means that our parameter space has been one-dimensional. Everything learned from data is incorporated into a single probability distribution representing the relative probabilities of different values for the rate θ.

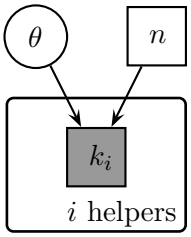
For many problems in cognitive science (and more generally), however, there will be more than one unknown variable of interest, and they will interact. A simple case of this general property is a binomial process in which both the rate θ *and* the total number n are unknown, and so the problem is to infer both simultaneously from counts of successes k.

Box 3.4 Today's posterior is tomorrow's prior

The idea that prior information about parameters can be transformed into posterior information, and hence prior predictive information about data can be transformed into posterior predictive information, can be continued indefinitely. As more information becomes available, usually as more data are collected, uncertainty about parameters and predictive distributions are naturally updated in the Bayesian approach.



The figure shows the incorporation of a sequence of data for the common model in Figure 3.10. Panel "a" shows the uniform prior over the common rate. Panel "b" shows the prior predictive, for the two counts of successes out of 10 trials. The gray cross corresponds to the observed data, which has yet to be incorporated, but can be compared to the prior predictive distribution. Panel "c" shows the posterior on the rate that now incorporates the data, and panel "d" shows the resulting posterior predictive. The first data are now shown by the black cross in this posterior predictive, since they are incorporated, but a new second data set, in the form of the different gray cross, is about to arrive. These new data are incorporated into the posterior distribution over the rate in panel "e," which leads to the posterior prediction in panel "f." And so the process can continue. Notice how the distribution over the rate parameter in panel "c" is the posterior distribution with respect to the first data set, but acts as the prior for the second data set. This leads to Lindley's Bayesian motto "Today's posterior is tomorrow's prior."



$$k_i \sim \text{Binomial}(\theta, n)$$
$$\theta \sim \text{Beta}(1,1)$$
$$n \sim \text{Categorical}(\underbrace{\frac{1}{n_{\max}}, \ldots, \frac{1}{n_{\max}}}_{m})$$

Fig. 3.12 Graphical model for the joint inference of n and θ from a set of m observed counts of successes $k_1, \ldots, k_m$.

To make the problem concrete, suppose there are five helpers distributing a bundle of surveys to houses. It is known that each bundle contained the same number of surveys, n, but the number itself is not known. The only available relevant information is that the maximum bundle is $n_{\max} = 500$, and so n must be between 1 and $n_{\max}$.

In this problem, it is also not known what the rate of return for the surveys is. But, it is assumed that each helper distributed to houses selected in a random enough way that it is reasonable to believe the return rates are the same. It is also assumed to be reasonable to set a uniform prior on this common rate $\theta \sim \text{Beta}\big(1,1\big)$.

Inferences can simultaneously be made about n and θ from the observed number of surveys returned for each of the helpers. Assuming the surveys themselves can be identified with their distributing helper when returned, the data will take the form of $m = 5$ counts, one for each helper, giving the number of returned surveys for each.

The graphical model for this problem is shown in Figure 3.12, and the script `Survey.txt` implements the graphical model in WinBUGS. Note the use of the categorical distribution, which gives probabilities to a finite set of nominal outcomes:

```
# Inferring Return Rate and Number of Surveys from Observed Returns
model{
  # Observed Returns
  for (i in 1:m){
     k[i] ~ dbin(theta,n)
  }
  # Priors on Rate Theta and Number n
  theta ~ dbeta(1,1)
  n ~ dcat(p[])
  for (i in 1:nmax){
    p[i] <- 1/nmax
  }
}
```

The code `Survey.m` or `Survey.R` uses the data $k = \{16, 18, 22, 25, 27\}$, and then calls WinBUGS to sample from the graphical model. Figure 3.13 shows the joint posterior distribution over n and θ as a scatter-plot, and the marginal distributions of each as histograms.

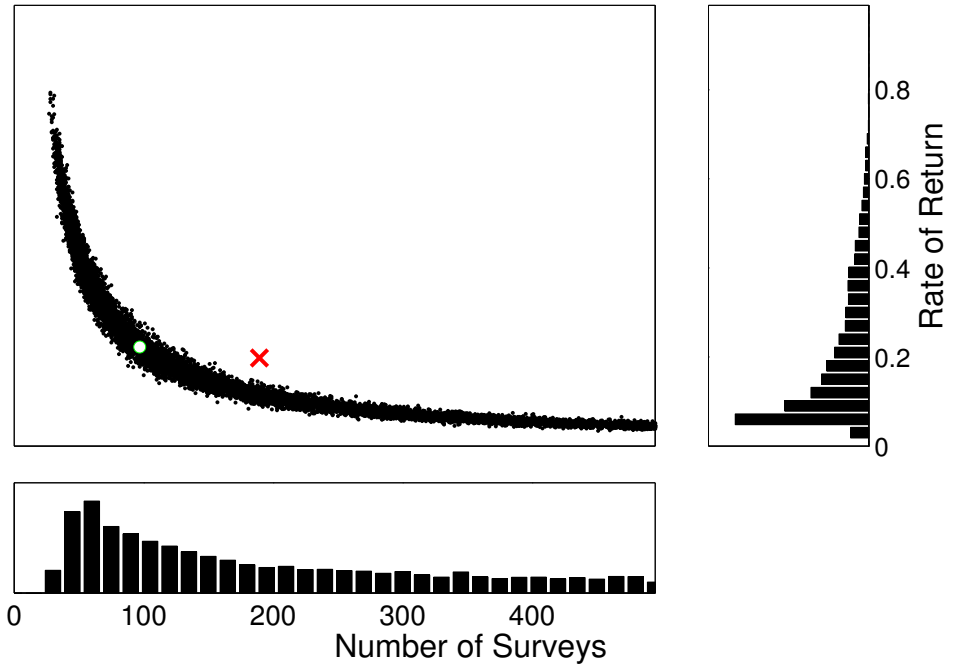


Fig. 3.13 Joint posterior distribution of the probability of return θ and the number of surveys n for $m = 5$ observed counts $k = \{16, 18, 22, 25, 27\}$. The histograms show the marginal densities. The cross shows the expected value of the joint posterior, and the circle shows the mode (i.e., maximum likelihood), both estimated from the posterior samples.

It is clear that the joint posterior distribution carries more information than the marginal posterior distributions. This is very important. It means that just looking at the marginal distributions will not give a complete account of the inferences made, and may provide a misleading account.

An intuitive graphical way to see that there is extra information in the joint posterior is to see if it is well approximated by the product of the marginal distributions. Imagine sampling a point from the histogram for n where there is non-negligible marginal density, such as at $n = 300$. Imagine also sampling points from the histogram for θ, where there is non-negligible marginal density, such as at $\theta = 0.4$. These choices correspond to a single point in the joint posterior density space. Now imagine repeating this process many times. It should be clear that the resulting scatter-plot would be different from the joint posterior scatter-plot in Figure 3.13. So, the joint distribution carries information not available from the marginal distributions.

For this example, it is intuitively obvious why the joint posterior distribution has the clear non-linear structure it does. One possible way in which 20 surveys might be returned is if there were only about 50 surveys, but 40% were returned. Another

possibility is that there were 500 surveys, but only a 4% return rate. In general, the number and return rate can trade-off against each other, sweeping out the joint posterior distribution seen in Figure 3.13.

Exercises

Exercise 3.6.1 The basic moral of this example is that it is often worth thinking about joint posterior distributions over model parameters. In this case the marginal posterior distributions are probably misleading. Potentially even more misleading are common (and often perfectly appropriate) point estimates of the joint distribution. The cross in Figure 3.13 shows the expected value of the joint posterior, as estimated from the samples. Notice that it does not even lie in a region of the parameter space with any posterior mass. Does this make sense?

Exercise 3.6.2 The circle in Figure 3.13 shows an approximation to the mode (i.e., the sample with maximum likelihood) from the joint posterior samples. Does this make sense?

Exercise 3.6.3 Try the very slightly changed data $k = \{16, 18, 22, 25, 28\}$. How does this change the joint posterior, the marginal posteriors, the expectation, and the mode? If you were comfortable with the mode, are you still comfortable?

Exercise 3.6.4 If you look at the sequence of samples in the trace plot, some autocorrelation is evident. The samples "sweep" through high and low values in a systematic way, showing the dependency of a sample on those immediately preceding. This is a deviation from the ideal situation in which posterior samples are independent draws from the joint posterior. Try thinning the sampling, taking only every 100th sample, by setting `nthin=100` in Matlab or `n.thin=100` in R. To make the computational time reasonable, reduce the number of samples collected after thinning to just 500 (i.e., run 50,000 total samples, so that 500 are retained after thinning). How is the sequence of samples visually different with thinning?[2]

[2] A note for R2jags users: at the time of writing, R2jags mistakenly randomizes the values in the `sims.array` object whenever you run a single chain. Until this error is fixed it is safest to run multiple chains, at least when you are interested in examining autocorrelation. See also the last few posts here: `http://sourceforge.net/p/mcmc-jags/discussion/610037/thread/cc61b820/?limit=50#83b4`.

4 Inferences with Gaussians

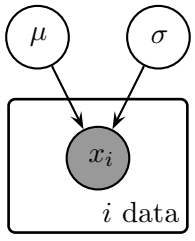
4.1 Inferring a mean and standard deviation

One of the most common inference problems involves assuming data following a Gaussian (also known as a Normal, Central, or Maxwellian) distribution, and inferring the mean and standard deviation of this distribution from a sample of observed independent data.

The graphical model representation for this problem is shown in Figure 4.1. The data are the n observations $x_1, \ldots, x_n$. The mean of the Gaussian is μ and the standard deviation is σ. WinBUGS parameterizes the Gaussian distribution in terms of the mean and precision, not the mean and variance or the mean and standard deviation. These are all simply related, with the variance being σ^2 and the precision being $\lambda = 1/\sigma^2$.

Here the prior used for μ is intended to be only weakly informative. That is, it is a prior intended to convey little information about the mean, so that inference will be primarily dependent upon relevant data. It is a Gaussian centered on zero, but with very low precision (i.e., very large variance), and gives prior probability to a wide range of possible means for the data. When the goal is to estimate parameters, this sort of approach is relatively non-controversial.

Setting priors for standard deviations (or variances, or precisions) is trickier, and certainly more controversial. If there is any relevant information that helps put the data on scale, so that bounds can be set on reasonable possibilities for the standard deviation, then setting a uniform over that range is advocated by Gelman (2006). In this first example, we assume the data are all small enough that setting an upper bound of 10 on the standard deviation covers all the possibilities.



$$\mu \sim \text{Gaussian}(0, 0.001)$$
$$\sigma \sim \text{Uniform}(0, 10)$$
$$x_i \sim \text{Gaussian}(\mu, \tfrac{1}{\sigma^2})$$

Fig. 4.1 Graphical model for inferring the mean and standard deviation of data generated by a Gaussian distribution.

The script `Gaussian.txt` implements the graphical model in WinBUGS. Note the conversion of the standard deviation `sigma` into the precision parameter `lambda` used to sample from a Gaussian:

```
# Inferring the Mean and Standard Deviation of a Gaussian
model{
  # Data Come From A Gaussian
  for (i in 1:n){
    x[i] ~ dnorm(mu,lambda)
  }
  # Priors
  mu ~ dnorm(0,.001)
  sigma ~ dunif(0,10)
  lambda <- 1/pow(sigma,2)
}
```

The code `Gaussian.m` or `Gaussian.R` creates some artificial data, and applies the graphical model to make inferences from data. The code does not produce a graph, or any other output. But all of the information you need to analyze the results is in the returned variables `samples` and `stats`.

Exercises

Exercise 4.1.1 Try a few data sets, varying what you expect the mean and standard deviation to be, and how many data you observe.

Exercise 4.1.2 Plot the *joint* posterior of μ and σ. That is, plot the samples from μ against those of σ. Interpret the shape of the joint posterior.

Exercise 4.1.3 Suppose you knew the standard deviation of the Gaussian was 1.0, but still wanted to infer the mean from data. This is a realistic question: For example, knowing the standard deviation might amount to knowing the noise associated with measuring some psychological trait using a test instrument. The x_i values could then be repeated measures for the same person, and their mean the trait value you are trying to infer. Modify the WinBUGS script and Matlab or R code to do this. What does the revised graphical model look like?

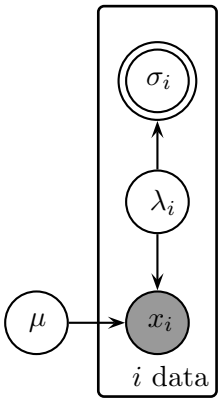
Exercise 4.1.4 Suppose you knew the mean of the Gaussian was zero, but wanted to infer the standard deviation from data. This is also a realistic question: Suppose you know that the error associated with a measurement is unbiased, so its average or mean is zero, but you are unsure how much noise there is in the instrument. Inferring the standard deviation is then a sensible way to infer the noisiness of the instrument. Once again, modify the WinBUGS script and Matlab or R code to do this. Once again, what does the revised graphical model look like?

4.2 The seven scientists

This problem is from MacKay (2003, p. 309) where it is, among other things, treated to a Bayesian solution, but not quite using a graphical modeling approach, nor relying on computational sampling methods.

Seven scientists with wildly-differing experimental skills all make a measurement of the same quantity. They get the answers $x = \{-27.020, 3.570, 8.191, 9.898, 9.603, 9.945, 10.056\}$. Intuitively, it seems clear that the first two scientists are pretty inept measurers, and that the true value of the quantity is probably just a bit below 10. The main problem is to find the posterior distribution over the measured quantity, telling us what we can infer from the measurement. A secondary problem is to infer something about the measurement skills of the seven scientists.

The graphical model for one way of solving this problem is shown in Figure 4.2. The assumption is that all the scientists have measurements that follow a Gaussian distribution, but with different standard deviations. However, because they are all measuring the same quantity, each Gaussian has the same mean, and it is just the standard deviation that differs.



$$\mu \sim \text{Gaussian}(0, 0.001)$$
$$\lambda_i \sim \text{Gamma}(0.001, 0.001)$$
$$\sigma_i \leftarrow 1/\sqrt{\lambda_i}$$
$$x_i \sim \text{Gaussian}(\mu, \lambda_i)$$

Fig. 4.2 Graphical model for the seven scientists problem.

Notice that we have used a different approach to assign priors to the standard deviations. The previous example, as shown in Figure 4.1, used a uniform distribution. The current example, shown in Figure 4.2, uses a gamma distribution for the priors on the precisions. This is another standard approach, which has some attractive theoretical motivations, but is critically discussed by Gelman (2006).

The script `SevenScientists.txt` implements the graphical model in Figure 4.2 in WinBUGS:

```
# The Seven Scientists
model{
  # Data Come From Gaussians With Common Mean But Different Precisions
  for (i in 1:n){
    x[i] ~ dnorm(mu,lambda[i])
```

Box 4.1 Priors on precisions

The practice of assigning $\mathrm{Gamma}(0.001, 0.001)$ priors on precision parameters is theoretically motivated by scale invariance arguments, meaning that priors are chosen so that changing the measurement scale of the data does not affect inference. The invariant prior on precision λ corresponds to a uniform distribution on $\log \sigma$, that is, $p(\sigma^2) \propto 1/\sigma^2$, or a $\mathrm{Gamma}(a \to 0, b \to 0)$ distribution. This invariant prior distribution, however, is *improper* (i.e., the area under the curve is unbounded), which means it is not really a distribution, but the limit of a sequence of distributions (see Jaynes, 2003). WinBUGS requires the use of proper distributions, and the $\mathrm{Gamma}(0.001, 0.001)$ prior is intended as a proper approximation to the theoretically motivated improper prior. This raises the issue of whether inference is sensitive to the essentially arbitrary value 0.001, and it is sometimes the case that using other small values such as 0.01 or 0.1 leads to more stable sampling in WinBUGS.

```
  }
  # Priors
  mu ~ dnorm(0,.001)
  for (i in 1:n){
    lambda[i] ~ dgamma(.001,.001)
    sigma[i] <- 1/sqrt(lambda[i])
  }
}
```

Notice that the graphical model implements the prior on the precisions, but also re-parameterizes to the standard deviation scale, which is often more easily interpretable.

The code `SevenScientists.m` or `SevenScientists.R` applies the seven scientist data to the graphical model.

Exercises

Exercise 4.2.1 Draw posterior samples using the Matlab or R code, and reach conclusions about the value of the measured quantity, and about the accuracies of the seven scientists.

Exercise 4.2.2 Change the graphical model in Figure 4.2 to use a uniform prior over the standard deviations, as was done in Figure 4.1. Experiment with the effect the upper bound of this uniform prior has on inference.

Box 4.2 **Ill-posed problems**

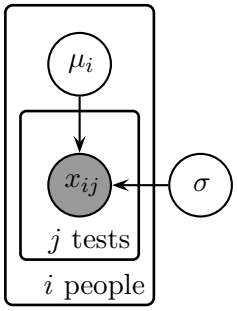
"If one fails to specify the prior information, a problem of inference is just as ill-posed as if one had failed to specify the data." (Jaynes, 2003, p. 373).

4.3 Repeated measurement of IQ

In this example, we consider how to estimate the IQ of a set of people, each of whom have done multiple IQ tests. The data are the measures x_{ij} for the $i = 1, \ldots, n$ people and their $j = 1, \ldots, m$ repeated test scores.

We assume that the differences in repeated test scores are distributed as Gaussian error terms with zero mean and unknown precision. The mean of the Gaussian of a person's test scores corresponds to their latent true IQ. This will be different for each person. The standard deviation of the Gaussians corresponds to the accuracy of the testing instruments in measuring the one underlying IQ value. We assume this is the same for every person, since it is conceived as a property of the tests themselves.

The graphical model for this problem is shown in Figure 4.3. Because we know quite a bit about the IQ scale, it makes sense to set priors for the mean and standard deviation using this knowledge. Our first attempt to set priors (these are revisited in the exercises) simply assume the actual IQ values are equally likely to be anywhere between 0 and 300, and standard deviations are anywhere between 0 and 100.



$$\mu_i \sim \text{Uniform}(0, 300)$$
$$\sigma \sim \text{Uniform}(0, 100)$$
$$x_{ij} \sim \text{Gaussian}(\mu_i, \tfrac{1}{\sigma^2})$$

Fig. 4.3 Graphical model for inferring the IQ from repeated measures.

The script `IQ.txt` implements the graphical model in WinBUGS:

```
# Repeated Measures of IQ
model{
  # Data Come From Gaussians With Different Means But Common Precision
  for (i in 1:n){
    for (j in 1:m){
      x[i,j] ~ dnorm(mu[i],lambda)
    }
  }
```

```
  # Priors
  sigma ~ dunif(0,100)
  lambda <- 1/pow(sigma,2)
  for (i in 1:n){
    mu[i] ~ dunif(0,300)
  }
}
```

The code `IQ.m` or `IQ.R` creates a data set corresponding to there being three people, with test scores of $(90, 95, 100)$, $(105, 110, 115)$, and $(150, 155, 160)$, and applies the graphical model.

Exercises

Exercise 4.3.1 Use the posterior distribution for each person's μ_i to estimate their IQ. What can we say about the precision of the IQ test?

Exercise 4.3.2 Now, use a more realistic prior assumption for the μ_i means. Theoretically, IQ distributions should have a mean of 100, and a standard deviation of 15. This corresponds to having a prior of `mu[i]` $\sim$ `dnorm(100,.0044)`, instead of `mu[i]` $\sim$ `dunif(0,300)`, because $1/15^2 = 0.0044$. Make this change in the WinBUGS script, and re-run the inference. How do the estimates of IQ given by the means change? Why?

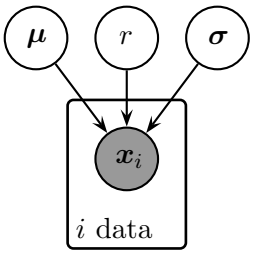
Exercise 4.3.3 Repeat both of the above stages (i.e., using both priors on μ_i) with a new, but closely related, data set that has scores of $(94, 95, 96)$, $(109, 110, 111)$, and $(154, 155, 156)$. How do the different prior assumptions affect IQ estimation for these data. Why does it not follow the same pattern as the previous data?

5 Some examples of data analysis

5.1 Pearson correlation

The Pearson product-moment correlation coefficient, usually denoted r, is a widely used measure of the relationship between two variables. It ranges from -1, indicating a perfect negative linear relationship, to $+1$, indicating a perfect positive relationship. A value of 0 indicates that there is no linear relationship. Usually the correlation r is reported as a single point estimate, perhaps together with a frequentist significance test.[1]

But, rather than just having a single number to measure the correlation, it would be nice to have a posterior distribution for r, saying how likely each possible level of correlation was. There are frequentist confidence interval methods that try to do this, as well as various analytic Bayesian results based on asymptotic approximations (e.g., Donner & Wells, 1986). An advantage of using a computational approach is the flexibility in the assumptions that can be made. It is possible to set up a graphical model that allows inferences about the correlation coefficient for any set of prior assumptions about the correlation.



$$
\begin{aligned}
\mu_1, \mu_2 &\sim \text{Gaussian}(0, 0.001) \\
\sigma_1, \sigma_2 &\sim \text{InvSqrtGamma}(0.001, 0.001) \\
r &\sim \text{Uniform}(-1, 1) \\
\boldsymbol{x}_i &\sim \text{MvGaussian}\left((\mu_1, \mu_2) , \begin{bmatrix} \sigma_1^2 & r\sigma_1\sigma_2 \\ r\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}^{-1} \right)
\end{aligned}
$$

Fig. 5.1 Graphical model for inferring a correlation coefficient.

One graphical model for doing this is shown in Figure 5.1. The observed data take the form $\boldsymbol{x}_i = (x_{i1}, x_{i2})$ for the ith observation, and, following the theory behind the correlation coefficient, are modeled as draws from a multivariate Gaussian distribution. The parameters of this distribution are the means $\boldsymbol{\mu} = (\mu_1, \mu_2)$ and standard deviations $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ of the two variables, and the correlation coefficient r that links them.

[1] Frequentist or orthodox statistics is familiar to all cognitive scientists. Key frequentist concepts include the p-value, power, confidence intervals, and Type-I error rate. We believe that for scientific inference, the frequentist approach is inefficient at best and misleading at worst.

Box 5.1 Frequentist subjectivity

"Today one wonders how it is possible that orthodox logic continues to be taught in some places year after year and praised as 'objective', while Bayesians are charged with 'subjectivity'. Orthodoxians, preoccupied with fantasies about nonexistent data sets and, in principle, unobservable limiting frequencies—while ignoring relevant prior information—are in no position to charge anybody with 'subjectivity'." (Jaynes, 2003, p. 550).

In Figure 5.1, the standard deviations are assigned relatively uninformative inverse-square-root-gamma distributions. This is equivalent to placing gamma distributions on precisions, as was done in the seven scientists example in Section 4.2. The correlation coefficient itself is given a uniform prior over its possible range. All of these choices would be easily modified, with one obvious possible change being to give the prior for the correlation more density around 0.

The script `Correlation_1.txt` implements the graphical model in WinBUGS:

```
# Pearson Correlation
model{
  # Data
  for (i in 1:n){
    x[i,1:2] ~ dmnorm(mu[],TI[,])
  }
  # Priors
  mu[1] ~ dnorm(0,.001)
  mu[2] ~ dnorm(0,.001)
  lambda[1] ~ dgamma(.001,.001)
  lambda[2] ~ dgamma(.001,.001)
  r ~ dunif(-1,1)
  # Reparameterization
  sigma[1] <- 1/sqrt(lambda[1])
  sigma[2] <- 1/sqrt(lambda[2])
  T[1,1] <- 1/lambda[1]
  T[1,2] <- r*sigma[1]*sigma[2]
  T[2,1] <- r*sigma[1]*sigma[2]
  T[2,2] <- 1/lambda[2]
  TI[1:2,1:2] <- inverse(T[1:2,1:2])
}
```

The code `Correlation_1.m` or `Correlation_1.R` includes two data sets. Both involve fabricated data comparing response times in a semantic verification task (e.g., "Is a whale a fish?") on the x-axis with IQ measures on the y-axis, looking for a correlation between simple measures of decision-making and general intelligence.

For the first data set in the Matlab and R code, the results shown in Figure 5.2 are produced. The left panel shows a scatter-plot of the raw data. The right panel shows the posterior distribution of r, together with the standard frequentist point estimate.

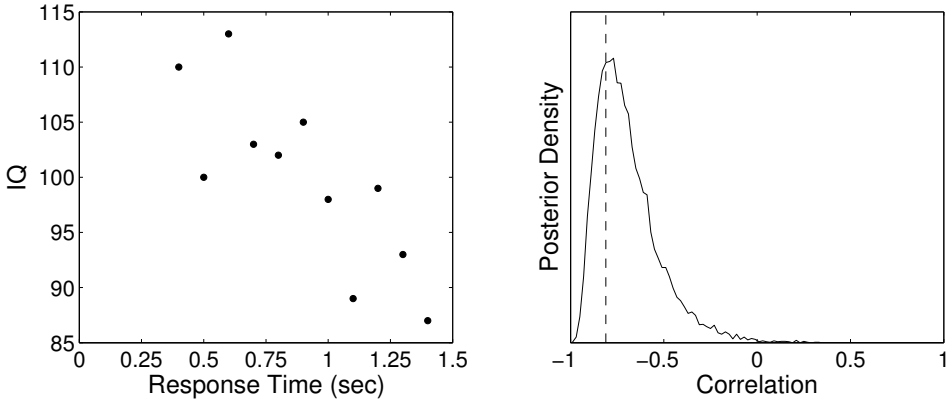


Fig. 5.2 Data (left panel) and posterior distribution for correlation coefficient (right panel). The broken line shows the frequentist point estimate.

Exercises

Exercise 5.1.1 The second data set in the Matlab and R code is just the first data set from Figure 5.2 repeated twice. Set `dataset=2` to consider these repeated data, and interpret the differences in the posterior distributions for r.

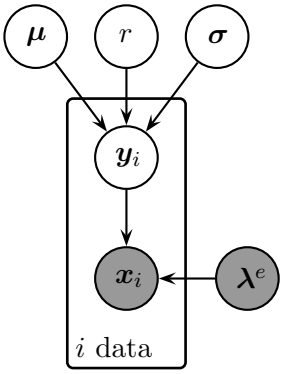
Exercise 5.1.2 Do you find the priors on μ_1 and μ_2 to be reasonable?

Exercise 5.1.3 The current graphical model assumes that the values from the two variables—the $\boldsymbol{x}_i = (x_{i1}, x_{i2})$—are observed with perfect accuracy. When might this be a problematic assumption? How could the current approach be extended to make more realistic assumptions?

5.2 Pearson correlation with uncertainty

We now tackle the problem asked by the last question in the previous section, and consider the correlations when there is uncertainty about the exact values of variables. It is likely that each individual response time is measured very accurately, since it is a physical quantity and good measurement tools exist. But the measurement of IQ seems likely to be less precise, since it is a psychological quantity, and measurement tools like IQ tests are less accurate. The uncertainty in measurement should be incorporated in an assessment of the correlation between the variables (e.g., Behseta, Berdyyeva, Olson, & Kass, 2009).

A simple approach for including this uncertainty is adopted by the graphical model in Figure 5.3. The observed data still take the form $\boldsymbol{x}_i = (x_{i1}, x_{i2})$ for the ith person's response time and IQ measure. But these observations are now sampled from a Gaussian distribution, centered on the unobserved true response time and IQ of that person, denoted $\boldsymbol{y}_i = (y_{i1}, y_{i2})$. These true values are then modeled as the



$$\mu_1, \mu_2 \sim \text{Gaussian}(0, 0.001)$$
$$\sigma_1, \sigma_2 \sim \text{InvSqrtGamma}(0.001, 0.001)$$
$$r \sim \text{Uniform}(-1, 1)$$
$$\boldsymbol{y}_i \sim \text{MvGaussian}\left((\mu_1, \mu_2), \begin{bmatrix} \sigma_1^2 & r\sigma_1\sigma_2 \\ r\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}^{-1}\right)$$
$$x_{ij} \sim \text{Gaussian}(y_{ij}, \lambda_j^e)$$

Fig. 5.3 Graphical model for inferring a correlation coefficient, when there is uncertainty inherent in the measurements.

$\boldsymbol{x}$ were in the previous model in Figure 5.1, as draws from a multivariate Gaussian distribution.

The precision of the measurements is captured by $\boldsymbol{\lambda}^e = (\lambda_1^e, \lambda_2^e)$ of the Gaussian draws for the observed data, $x_{ij} \sim \text{Gaussian}(y_{ij}, \lambda_j^e)$. The graphical model in Figure 5.3 assumes that these precisions are known.

The script `Correlation_2.txt` implements the graphical model shown in WinBUGS:

```
# Pearson Correlation With Uncertainty in Measurement
model{
  # Data
  for (i in 1:n){
    y[i,1:2] ~ dmnorm(mu[],TI[,])
    for (j in 1:2){
      x[i,j] ~ dnorm(y[i,j],lambdaerror[j])
    }
  }
  # Priors
  mu[1] ~ dnorm(0,.001)
  mu[2] ~ dnorm(0,.001)
  lambda[1] ~ dgamma(.001,.001)
  lambda[2] ~ dgamma(.001,.001)
  r ~ dunif(-1,1)
  # Reparameterization
  sigma[1] <- 1/sqrt(lambda[1])
  sigma[2] <- 1/sqrt(lambda[2])
  T[1,1] <- 1/lambda[1]
  T[1,2] <- r*sigma[1]*sigma[2]
  T[2,1] <- r*sigma[1]*sigma[2]
  T[2,2] <- 1/lambda[2]
  TI[1:2,1:2] <- inverse(T[1:2,1:2])
}
```

The code `Correlation_2.m` or `Correlation_2.R` uses the same data as in the previous section, but has different analyses because of the different assumptions about the uncertainty in measurement. In these new analyses, we assume that

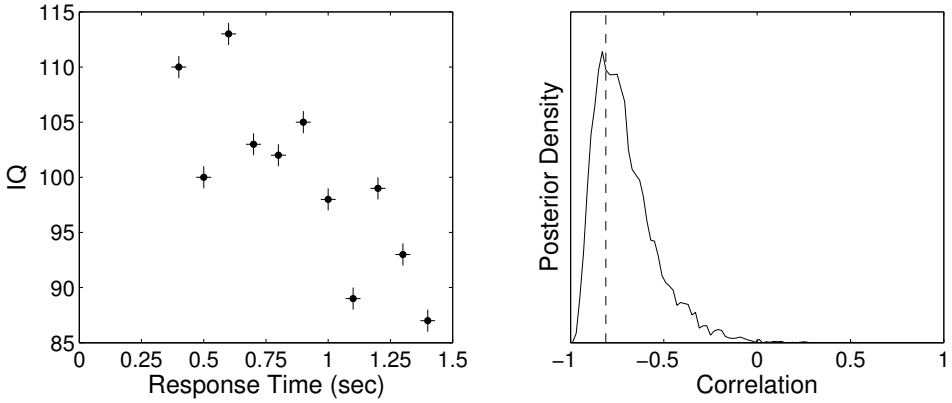


Fig. 5.4 Data (left panel), including error bars showing uncertainty in measurement, and posterior distribution for the correlation coefficient (right panel). The broken line shows the frequentist point estimate.

measurement uncertainty is originally expressed in terms of standard deviations, and then re-parameterized and supplied to the graphical model as precisions. The specific assumption is that $\sigma_1^e = .03$ for response times (which seem likely to be measured accurately) and $\sigma_2^e = 1$ for IQ (which seems near the smallest plausible value, so we assume that IQ is also measured accurately). The results of these assumptions using the model are shown in Figure 5.4. The left panel shows a scatterplot of the raw data, together with error bars representing the uncertainty quantified by the assumed standard deviations σ_1^e and σ_2^e. The right panel shows the posterior distribution of r, together with the standard frequentist point estimate.

Exercises

Exercise 5.2.1 Compare the results obtained in Figure 5.4 with those obtained earlier using the same data, in Figure 5.2, for the model without any account of uncertainty in measurement.

Exercise 5.2.2 Generate results for the second data set, which changes $\sigma_2^e = 10$ for the IQ measurement. Compare these results with those obtained assuming $\sigma_2^e = 1$.

Exercise 5.2.3 The graphical model in Figure 5.3 assumes the uncertainty for each variable is known. How could this assumption be relaxed to the case where the uncertainty is unknown?

Exercise 5.2.4 The graphical model in Figure 5.3 assumes the uncertainty for each variable is the same for all observations. How could this assumption be relaxed to the case where, for example, extreme IQs are less accurately measured than IQs in the middle of the standard distribution?

5.3 The kappa coefficient of agreement

An important statistical inference problem in a range of physical, biological, behavioral, and social sciences is to decide how well one decision-making method agrees with another. An interesting special case considers only binary decisions, and views one of the decision-making methods as giving objectively true decisions to which the other aspires. This problem occurs often in medicine, when cheap or easily administered methods for diagnosis are evaluated in terms of how well they agree with a more expensive or complicated "gold standard" method.

For this problem, when both decision-making methods make n independent assessments, the data $\boldsymbol{y}$ take the form of four counts: a observations where both methods decide "one," b observations where the objective method decides "one" but the surrogate method decides "zero," c observations where the objective method decides "zero" but the surrogate method decides "one," and d observations where both methods decide "zero," with $n = a + b + c + d$.

A variety of orthodox statistical measures have been proposed for assessing agreement using these data (but see Basu, Banerjee, & Sen, 2000; Broemeling, 2009, for Bayesian approaches). Useful reviews are provided by Agresti (1992), Banerjee, Capozzoli, McSweeney, and Sinha (1999), Fleiss, Levin, and Paik (2003), Kraemer (1992), Kraemer, Periyakoil, and Noda (2004), and Shrout (1998). Of all the measures, however, it is reasonable to argue that the conclusion of Uebersax (1987) that "the kappa coefficient is generally regarded as the statistic of choice for measuring agreement" (p. 140) remains true.

Cohen's (1960) kappa statistic estimates the level of observed agreement

$$p_o = \frac{a+d}{n}$$

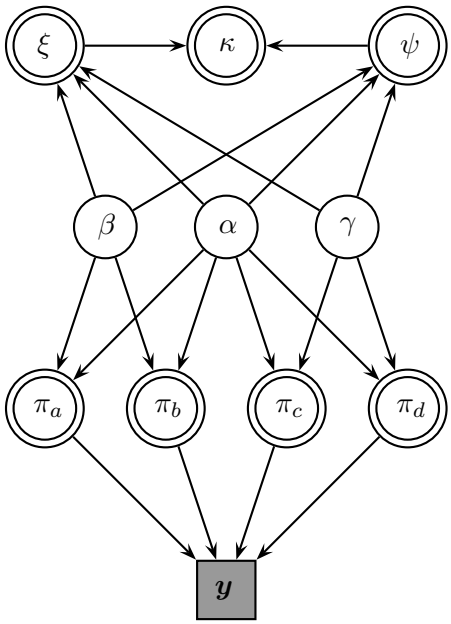
relative to the agreement that would be expected by chance alone (i.e., the overall probability for the first method to decide "one" times the overall probability for the second method to decide "one," and added to this the overall probability for the second method to decide "zero" times the overall probability for the first method to decide "zero")

$$p_e = \frac{(a+b)(a+c)+(b+d)(c+d)}{n^2},$$

and is given by

$$\kappa = \frac{p_o - p_e}{1 - p_e}.$$

Kappa lies on a scale of -1 to $+1$, with values below 0.4 often interpreted as "poor" agreement beyond chance, values between 0.4 and 0.75 interpreted as "fair to good" agreement beyond chance, and values above 0.75 interpreted as "excellent" agreement beyond chance (Landis & Koch, 1977). The key insight of kappa as a measure of agreement is its correction for chance agreement.



$$
\begin{aligned}
\kappa &\leftarrow (\xi - \psi) \,/\, (1 - \psi) \\
\xi &\leftarrow \alpha\beta + (1 - \alpha)\,\gamma \\
\psi &\leftarrow (\pi_a + \pi_b)\,(\pi_a + \pi_c) \\
&\quad + (\pi_b + \pi_d)\,(\pi_c + \pi_d) \\
\alpha, \beta, \gamma &\sim \text{Beta}(1, 1) \\
\pi_a &\leftarrow \alpha\beta \\
\pi_b &\leftarrow \alpha\,(1 - \beta) \\
\pi_c &\leftarrow (1 - \alpha)\,(1 - \gamma) \\
\pi_d &\leftarrow (1 - \alpha)\,\gamma \\
\boldsymbol{y} &\sim \text{Multinomial}(\left[\pi_a, \pi_b, \pi_c, \pi_d\right], n)
\end{aligned}
$$

Fig. 5.5 Graphical model for inferring the kappa coefficient of agreement.

The graphical model for a Bayesian version of kappa is shown in Figure 5.5. The key latent variables are α, β, and γ. The rate α is the rate at which the gold standard method decides "one." This means $(1 - \alpha)$ is the rate at which the gold standard method decides "zero." The rate β is the rate at which the surrogate method decides "one" when the gold standard also decides "one." The rate γ is the rate at which the surrogate method decides "zero" when the gold standard decides "zero." The best way to interpret β and γ is that they are the rate of agreement of the surrogate method with the gold standard, for the "one" and "zero" decisions respectively.

Using the rates α, β, and γ, it is possible to calculate the probabilities that both methods will decide "one," $\pi_a = \alpha\beta$, that the gold standard will decide "one" but the surrogate will decide "zero," $\pi_b = \alpha\,(1 - \beta)$, the gold standard will decide "zero" but the surrogate will decide "one," $\pi_c = (1 - \alpha)\,(1 - \gamma)$, and that both methods will decide "zero," $\pi_d = (1 - \alpha)\,\gamma$.

These probabilities, in turn, describe how the observed data, $\boldsymbol{y}$, made up of the counts a, b, c, and d, are generated. They come from a multinomial distribution with n trials, where on each trial there is a π_a probability of generating an a count, π_b probability for a b count, and so on.

So, observing the data $\boldsymbol{y}$ allows inferences to be made about the key rates α, β, and γ. The remaining variables in the graphical model in Figure 5.5 just re-express these rates in the way needed to provide an analogue to the kappa measure of chance-corrected agreement. The ξ variable measures the rate of agreement, which

is $\xi = \alpha\beta + (1-\alpha)\gamma$. The ψ variable measures the rate of agreement that would occur by chance, which is $\psi = (\pi_a + \pi_b)(\pi_a + \pi_c) + (\pi_b + \pi_d)(\pi_c + \pi_d)$, and could be expressed in terms of α, β, and γ. Finally κ is the chance-corrected measure of agreement on the -1 to $+1$ scale, given by $\kappa = (\xi - \psi)/(1 - \psi)$.

The script `Kappa.txt` implements the graphical model in WinBUGS:

```
# Kappa Coefficient of Agreement
model{
  # Underlying Rates
  # Rate Objective Method Decides "one"
  alpha ~ dbeta(1,1)
  # Rate Surrogate Method Decides "one" When Objective Method Decides "one"
  beta ~ dbeta(1,1)
  # Rate Surrogate Method Decides "zero" When Objective Method Decides "zero"
  gamma ~ dbeta(1,1)
  # Probabilities For Each Count
  pi[1] <- alpha*beta
  pi[2] <- alpha*(1-beta)
  pi[3] <- (1-alpha)*(1-gamma)
  pi[4] <- (1-alpha)*gamma
  # Count Data
  y[1:4] ~ dmulti(pi[],n)
  # Derived Measures
  # Rate Surrogate Method Agrees With the Objective Method
  xi <- alpha*beta+(1-alpha)*gamma
  # Rate of Chance Agreement
  psi <- (pi[1]+pi[2])*(pi[1]+pi[3])+(pi[2]+pi[4])*(pi[3]+pi[4])
  # Chance-Corrected Agreement
  kappa <- (xi-psi)/(1-psi)
}
```

The code `Kappa.m` or `Kappa.R` includes several data sets, described in the exercises below, for WinBUGS to sample from the graphical model.

Exercises

Exercise 5.3.1 *Influenza Clinical Trial.* Poehling, Griffin, and Dittus (2002) reported data evaluating a rapid bedside test for influenza using a sample of 233 children hospitalized with fever or respiratory symptoms. Of the 18 children known to have influenza, the surrogate method identified 14 and missed 4. Of the 215 children known not to have influenza, the surrogate method correctly rejected 210 but falsely identified 5. These data correspond to $a = 14$, $b = 4$, $c = 5$, and $d = 210$. Examine the posterior distributions of the interesting variables, and reach a scientific conclusion. That is, pretend you are a consultant for the clinical trial. What would your two- or three-sentence "take home message" conclusion be to your customers?

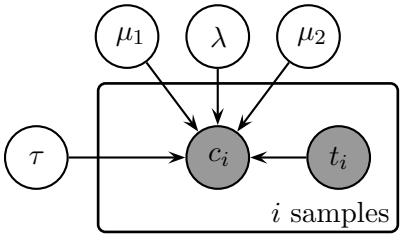
Exercise 5.3.2 *Hearing Loss Assessment Trial.* Grant (1974) reported data from a screening of a pre-school population intended to assess the adequacy of a school nurse assessment in relation to expert assessment of hearing loss. Of those children assessed by the expert as having hearing loss, 20 were correctly identified by the nurse and 7 were missed. Of those assessed by the expert

as not having hearing loss, 417 were correctly diagnosed by the nurse but 103 were incorrectly diagnosed as having hearing loss. These data correspond to $a = 20$, $b = 7$, $c = 103$, $d = 417$. Once again, examine the posterior distributions of the interesting variables, and reach a scientific conclusion. Once again, what would your two- or three-sentence "take home message" conclusion be to your customers?

Exercise 5.3.3 *Rare Disease.* Suppose you are testing a cheap instrument for detecting a rare medical condition. After 170 patients have been screened, the test results show that 157 did not have the condition, but 13 did. The expensive ground-truth assessment subsequently revealed that, in fact, none of the patients had the condition. These data correspond to $a = 0$, $b = 0$, $c = 13$, $d = 157$. Apply the kappa graphical model to these data, and reach a conclusion about the usefulness of the cheap instrument. What is special about this data set, and what does it demonstrate about the Bayesian approach?

5.4 Change detection in time series data

This case study involves near-infrared spectrographic data, in the form of oxygenated hemoglobin counts of frontal lobe activity during an attention task in Attention Deficit Hyperactivity Disorder (ADHD) adults. The interesting modeling problem is that a change is expected in the time series of counts because of the attention task. The statistical problem is to identify the change. To do this, we are going to make a number of strong assumptions. In particular, we will assume that the counts come from a Gaussian distribution that always has the same variance, but changes its mean at one specific point in time. The main interest is therefore in making an inference about this change point.



$$
\begin{aligned}
\mu_1, \mu_2 &\sim \text{Gaussian}(0, 0.001) \\
\lambda &\sim \text{Gamma}(0.001, 0.001) \\
\tau &\sim \text{Uniform}(0, t_{\max}) \\
c_i &\sim \begin{cases} \text{Gaussian}(\mu_1, \lambda) & \text{if } t_i < \tau \\ \text{Gaussian}(\mu_2, \lambda) & \text{if } t_i \geq \tau \end{cases}
\end{aligned}
$$

Fig. 5.6 Graphical model for detecting a single change point in time series.

Figure 5.6 presents a graphical model for detecting the change point. The observed data are the counts c_i at time t_i for the ith sample. The unobserved variable τ is the time at which the change happens, which controls whether the counts have mean μ_1 or μ_2. A uniform prior over the full range of possible times is assumed for

the change point, and generic weakly informative priors are given to the means and the precision.

The script `ChangeDetection.txt` implements this graphical model in WinBUGS:

```
# Change Detection
model{
  # Data Come From A Gaussian
  for (i in 1:n){
    c[i] ~ dnorm(mu[z1[i]],lambda)
  }
  # Group Means
  mu[1] ~ dnorm(0,.001)
  mu[2] ~ dnorm(0,.001)
  # Common Precision
  lambda  ~ dgamma(.001,.001)
  sigma  <- 1/sqrt(lambda)
  # Which Side is Time of Change Point?
  for (i in 1:n){
    z[i]  <- step(t[i]-tau)
    z1[i] <- z[i]+1
  }
  # Prior On Change Point
  tau ~ dunif(0,n)
}
```

Note the use of the `step` function. This function returns 1 if its argument is greater than or equal to zero, and 0 otherwise. The `z1` variable, however, serves as an indicator variable for `mu`, and therefore it needs to take on values 1 and 2. This is the reason `z` is transformed to `z1`. Study this code and make sure you understand what the `step` function accomplishes in this example.

The code `ChangeDetection.m` or `ChangeDetection.R` applies the model to the near-infrared spectrographic data. Uniform sampling is assumed, so that $t = 1, \ldots, 1178$.

The code produces a simple analysis, finding the mean of the posteriors for τ, μ_1 and μ_2, and using these summary points to overlay the inferences over the raw data. The result looks something like Figure 5.7. The time series data themselves are shown by the jagged black lines. The expected value of the posterior mean for the pre- and post-change levels, given by the posterior means for μ_1 and μ_2, are shown by the horizontal lines. The expected change point, given by the posterior mean for τ, is just under 800 samples, and is used to separate the plotting of the pre-change level from the post-change level.

Exercises

Exercise 5.4.1 Draw the posterior distributions for the change point, the means, and the common standard deviation.

Exercise 5.4.2 Figure 5.7 shows the mean of the posterior distribution for the change point (this is the point in time where the two horizontal lines meet). Can you think of a situation in which such a plotting procedure can be misleading?

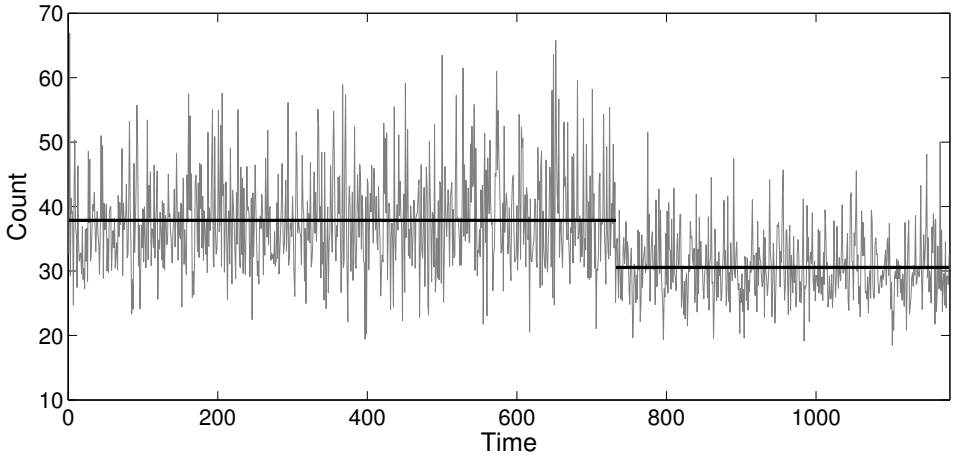


Fig. 5.7 Identification of a change point in time series data. The time series are shown by the jagged lines (note that these are observed data to be modeled; they are not chains from MCMC sampling), and the pre- and post-change levels around the expected change point are shown by the two overlaid horizontal lines.

Exercise 5.4.3 Imagine that you apply this model to a data set that has two change points instead of one. What could happen?

5.5 Censored data

Starting April 13 2005, Cha Sa-soon, a 68-year-old grandmother living in Jeonju, South Korea, repeatedly tried to pass the written exam for a driving license. In South Korea, this exam features 50 four-choice questions. In order to pass, one requires a score of at least 60 points out of a maximum of 100. Accordingly, we assume that each correct answer is worth two points, so that in order to pass, one needs to answer at least 30 questions correctly.

What has made Cha Sa-soon something of a national celebrity is that she failed to pass the test on 949 consecutive occasions, spending the equivalent of 4200 US dollars on application fees. In her last, 950th attempt, Cha Sa-soon scored the required minimum of 30 correct questions and finally passed her written exam. After her 775th failure, in February 2009, Mrs Cha told Reuters news agency, "I believe you can achieve your goal if you persistently pursue it. So don't give up your dream, like me. Be strong and do your best."

We know that on her final and 950th attempt, Cha Sa-soon answered 30 questions correctly. In addition, news agencies report that in her 949 unsuccessful attempts, the number of correct answers had ranged from 15 to 25. Armed with this knowledge, what can we say about θ, the latent probability that Cha Sa-soon can answer

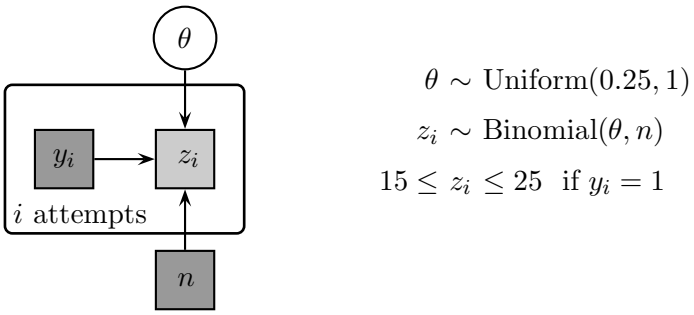


Fig. 5.8 Graphical model for inferring a rate from observed and censored data.

any one question correctly? Note that we assume each question is equally difficult, and that Cha Sa-soon does not learn from her earlier attempts.

The Cha Sa-soon data are special because we do not know the precise scores for the failed attempts. We only know that these scores range from 15 to 25. In statistical terms, these data are said to be censored, both from below and from above. We follow an approach inspired by Gelman and Hill (2007, p. 405) to apply WinBUGS to the problem of dealing with censored data.

Figure 5.8 presents a graphical model for dealing with the censored data. The variable z_i represents both the first 949 unobserved, and the final observed attempt. This means z_i is observed once, but not observed the other times. This sort of variable is known as *partially observed*, and is denoted in the graphical model by a lighter shading, between the dark shading of fully observed nodes, and the lack of shading for fully unobserved or latent nodes.

The variable y_i is a simple binary indicator variable, denoting whether or not the ith attempt is observed. The bounds $z^{\text{lo}} = 15$ and $z^{\text{hi}} = 25$ give the known censored interval for the unobserved attempts. Finally, $n = 50$ is the number of questions in the test. This means that $z_i \sim \text{Binomial}\big(\theta, n\big)_{\mathcal{I}(z^{\text{lo}}, z^{\text{hi}})}$ when y_i indicates a censored attempt, but that z_i is not censored for the final known score $z_{950} = 30$. The probability of a correct answer to a question, θ, is given a uniform prior between 0.25 and 1, corresponding to the assumption that chance accuracy of 1 in 4 is the lowest possible probability.

The script `ChaSaSoon.txt` implements this graphical model in WinBUGS:

```
# ChaSaSoon Censored Data
model{
  for (i in 1:nattempts){
    # If the Data Were Unobserved y[i]=1, Otherwise y[i]=0
    z.low[i]  <- 15*equals(y[i],1)+0*equals(y[i],0)
    z.high[i] <- 25*equals(y[i],1)+n*equals(y[i],0)
    z[i] ~ dbin(theta,n)I(z.low[i],z.high[i])
  }
  # Uniform Prior on Rate Theta
  theta ~ dbeta(1,1)I(.25,1)
}
```

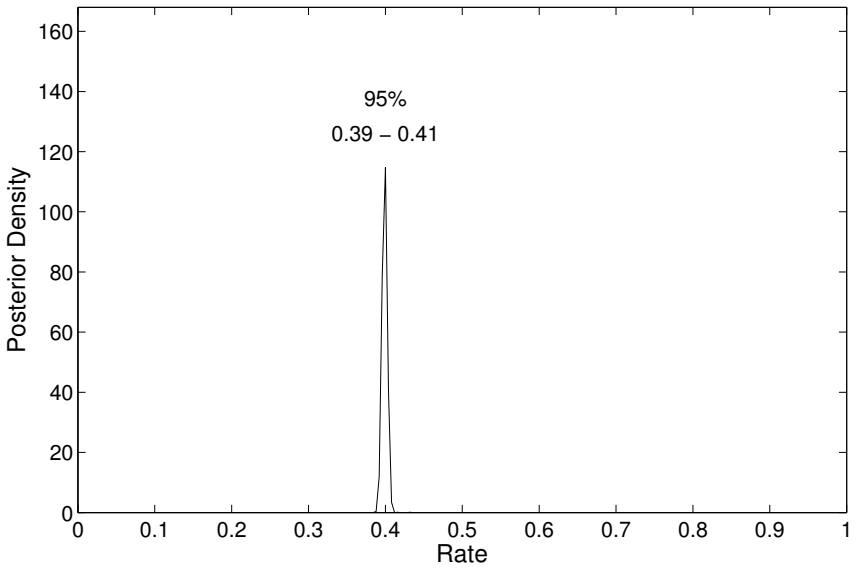


Fig. 5.9 Posterior density for Cha Sa-soon's rate of answering questions correctly.

Note the use of the `equals` command, which returns 1 when its arguments match, and 0 when they mismatch. Thus, when `y[i]=1`, for censored data, `z.low[i]` is set to 15 and `z.hi[i]` is set to 25. When `y[i]=0`, `z.low[i]` is set to 0 and `z.hi[i]` is set to n. These `z.low[i]` and `z.hi[i]` values are then applied to censor the binomial distribution that generates the test scores, using the WinBUGS `I` ("interval") command. In this way, the use of `equals` implements what might be considered the "case" or "if-then-else" logic of the model.

The code `ChaSaSoon.m` or `ChaSaSoon.R` applies the model to the data from Cha Sa-soon.[2] The posterior density for θ is shown in Figure 5.9, and can be seen to be relatively peaked. Despite the fact that we do not know the actual scores for 949 of the 950 results, we are still able to infer a lot about θ.

Exercises

Exercise 5.5.1 Do you think Cha Sa-soon could have passed the test by just guessing?

Exercise 5.5.2 What happens when you increase the interval in which you know the data are located, from 15–25 to something else?

Exercise 5.5.3 What happens when you decrease the number of failed attempts?

Exercise 5.5.4 What happens when you increase Cha Sa-soon's final score from 30?

[2] On some computers, WinBUGS will persistently return the mysterious error message "value of binomial `z[950]` must be greater than lower bound." If you know how to fix this error, we would love to hear from you. Otherwise, we can only suggest you run the code on a different computer.

Exercise 5.5.5 Do you think the assumption that all of the scores follow a binomial distribution with a single rate of success is a good model for these data?

5.6 Recapturing planes

An interesting inference problem that occurs in a number of fields is to estimate the size of a population, when a census is impossible, but repeated surveying is possible. For example, the goal might be to estimate the number of animals in a large woodland area that cannot be searched exhaustively. Or, the goal might be to decide how many students are on a campus, but it is not possible to count them all. Or, the goal might be to find out how many words in a given language a person knows, but it is not feasible to ask the person to list them all.

A clever sampling approach to this problem is given by capture-and-recapture methods. The basic idea is to capture (i.e., identify, tag, or otherwise remember) a sample at one time point, and then collect another sample. The number of items in the second sample that were also in the first then provides relevant information as to the population size. High recapture counts suggest that the population is small, and low recapture counts suggest that the population is large.

Probably the simplest possible version of this approach can be formalized with t as the unknown population size, x as the size of the first sample (i.e., number of units captured), and n as the size of the second sample from which a subset of k units were also present in the first sample (i.e., number of units recaptured). That is, first x animals are tagged or people remembered or words produced, then k out of n are seen again when a second sample is taken.

The statistical model to relate the counts and make inferences about the population size t is based on the hypergeometric distribution. The probability of seeing k items recaptured in a sample of size n, from the x originally captured in a population of size t, is

$$\Pr\left(K = k\right) = \binom{x}{k}\binom{t-x}{n-k}\Big/\binom{t}{n}.$$

Intuitively, the second sample involves taking n items from a population of t, and has k out of x recaptures, and $n-k$ other items out of the other $t-x$ in the population. Another way to formalize this is to say that the number of recaptures k is a sample from a hypergeometric distribution

$$k \sim \text{Hypergeometric}\big(n, x, t\big).$$

To make these ideas concrete, consider the challenge of estimating how many aircraft a small airline company has in its fleet. One day at an airport, you see 10 of the airline company's planes parked at adjacent gates, and record their unique identifying tail numbers. A few days later, at a different airport, you see 5 of the same company's planes. Looking at the tail number of those planes, you observe

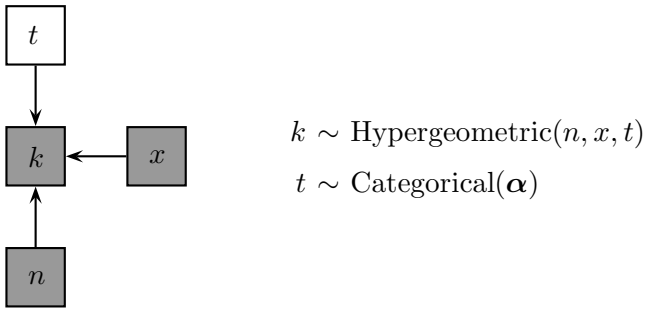


Fig. 5.10 Graphical model for inferring a population from capture-and-recapture data.

that 4 of the 5 were part of your original list. This is a capture-and-recapture problem with $x = 10$, $k = 4$, and $n = 5$.

The Bayesian approach to this problem involves assigning a prior to t, and using the hypergeometric distribution as the appropriate likelihood function. Conceptually, this means $k \sim \text{Hypergeometric}\big(n, x, t\big)$, as in the graphical model in Figure 5.10. The vector $\boldsymbol{\alpha}$ allows for any sort of prior mass to be given to all the possible counts for the population total. Since $x+(n-k)$ items are known to exist, one reasonable choice of prior might be to make every possibility from $x+(n-k)$ to $t^{\max}$ equally likely, where $t^{\max}$ is a sensible upper bound on the possible population. Suppose, for example, in the airplane problem that you know that the maximum number the company could possibly have is 50 planes, so that $t^{\max} = 50$.

While it is simple conceptually, there is a difficulty in implementing the graphical model in Figure 5.10. The problem is that WinBUGS does not provide the hypergeometric distribution. It is, however, possible to implement distributions that are not provided, but for which the likelihood function can be expressed in WinBUGS. This can be done using either the so-called "ones trick" or the "zeros trick."[3] These tricks rely on simple properties of the Poisson and Bernoulli distributions. By implementing the likelihood function of the new distribution within the Poisson or Bernoulli distribution, and forcing values of 1 or 0 to be sampled, it can be shown that the samples actually generated will come from the desired distribution.

The script `Planes.txt` implements the graphical model in Figure 5.10 in WinBUGS, using the zeros trick. Note how the terms in the log-likelihood expression for the hypergeometric distribution are built up to define `phi`, and a constant `C` is used to ensure the Poisson distribution is used with a positive value:

```
# Planes
model{
  # Hypergeometric Likelihood Via Zeros Trick
  logterm1 <- logfact(x)-logfact(k)-logfact(x-k)
  logterm2 <- logfact(t-x)-logfact(n-k)-logfact((t-x)-(n-k))
  logterm3 <- logfact(t)-logfact(n)-logfact(t-n)
  C <- 1000
```

[3] Using the zeros trick or ones trick in JAGS involves putting the assignment of `zeros` or `ones` inside the data definition block, rather than inside the model definition block.

Box 5.2 The zeros trick, ones trick, and WBDev

The zeros trick and ones trick are extremely useful, and relatively easy to implement in many cases, but a little difficult to understand conceptually. The key insight is that the negative log-likelihood of a sample of 0 from $\text{Poisson}(\phi)$ is ϕ, and similarly for a sample of 1 from $\text{Bernoulli}(\theta)$ it is θ. So, by setting $\log \phi$ or θ appropriately, and forcing 1 or 0 to be observed, sampling effectively proceeds from the distribution defined by ϕ or θ.

More complicated extensions to the distributions and functions available in WinBUGS require using the WinBUGS Development Interface (WBDev: Lunn, 2003). This is an add-on program that allows the user to hand-code functions and distributions in Component Pascal. Wetzels, Lee, and Wagenmakers (2010) provide a tutorial on WBDev that includes simple worked examples of defining new distributions and functions. More detailed cognitive science applications are provided by Wetzels, Vandekerckhove, et al. (2010) implementing the Expectancy-Valence model of decision-making as a function in WBDev, and Vandekerckhove et al. (2011) implementing the drift-diffusion model as a distribution in WBDev. Both of these applications would be impractical without WBDev.

```
  phi <- -(logterm1+logterm2-logterm3)+C
  zeros <- 0
  zeros ~ dpois(phi)
  # Prior on Population Size
  for (i in 1:tmax){
    tptmp[i] <- step(i-(x+n-k))
    tp[i] <- tptmp[i]/sum(tptmp[1:tmax])
  }
  t ~ dcat(tp[])
}
```

The code `Planes.m` or `Planes.R` applies the model to the data $x = 10$, $k = 4$, and $n = 5$, using uniform prior mass for all possible sizes between $x + (n - k) = 11$ and $t^{\max} = 50$. The posterior distribution for t is shown in Figure 5.11. The inference is that it is mostly likely there are not many more than 11 planes, which makes intuitive sense, since 4 out of 5 in the second sample were from the original set of 10.

Exercises

Exercise 5.6.1 Try changing the number of planes seen again in the second sample from $k = 4$ to $k = 0$. What inference do you draw about the population size now?

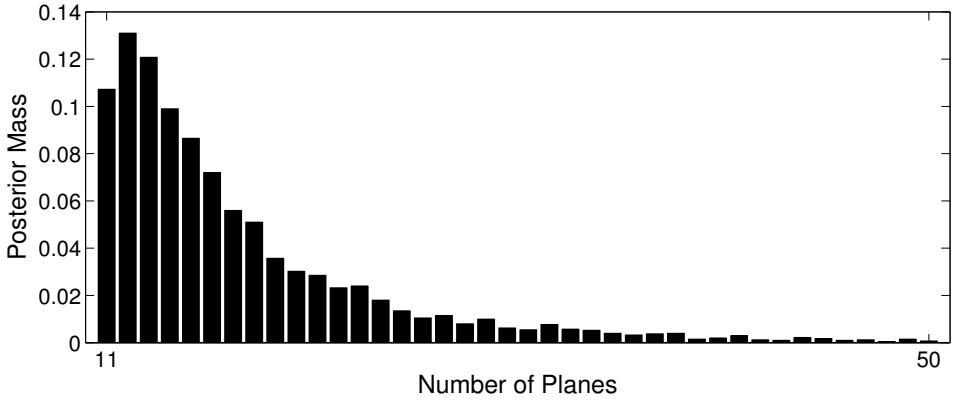


Fig. 5.11 Posterior mass for the number of planes, known to be 50 or fewer, based on a capture-recapture experiment with $x = 10$ planes in the first sample, and $k = 4$ out of $n = 5$ seen again in the second sample.

Exercise 5.6.2 How much impact does the upper bound $t^{\max} = 50$ have on the final conclusions when $k = 4$ and when $k = 0$? Develop your answer by trying both the $k = 4$ and $k = 0$ cases with $t^{\max} = 100$.

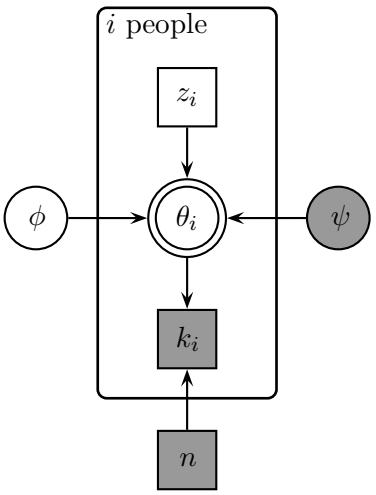
Exercise 5.6.3 Suppose, having obtained the posterior mass in Figure 5.11, the same fleet of planes was subjected to a new sighting at a different airport at a later day. What would be an appropriate prior for t?

6 Latent-mixture models

6.1 Exam scores

Suppose a group of 15 people sit an exam made up of 40 true-or-false questions, and they get 21, 17, 21, 18, 22, 31, 31, 34, 34, 35, 35, 36, 39, 36, and 35 right. These scores suggest that the first 5 people were just guessing, but the last 10 had some level of knowledge.

One way to make statistical inferences along these lines is to assume there are two different groups of people. These groups have different probabilities of success, with the guessing group having a probability of 0.5, and the knowledge group having a probability greater than 0.5. Whether each person belongs to the first or the second group is a latent or unobserved variable that can take just two values. Using this approach, the goal is to infer to which group each person belongs, and also the rate of success for the knowledge group.



$$
\begin{aligned}
z_i &\sim \text{Bernoulli}(0.5) \\
\phi &\sim \text{Uniform}(0.5, 1) \\
\psi &\leftarrow 0.5 \\
\theta_i &\leftarrow \begin{cases} \phi \text{ if } z_i = 1 \\ \psi \text{ if } z_i = 0 \end{cases} \\
k_i &\sim \text{Binomial}(\theta_i, n)
\end{aligned}
$$

Fig. 6.1 Graphical model for inferring membership of two latent groups, with different rates of success in answering exam questions.

A graphical model for doing this is shown in Figure 6.1. The number of correct answers for the ith person is k_i, and is out of $n = 40$. The probability of success on each question for the ith person is the rate θ_i. This rate is either ψ, if the person is

in the guessing group, or ϕ if the person is in the knowledge group. Which group the ith person belongs to is determined by a binary indicator variable z_i, with $z_i = 0$ if the ith person is in the guessing group, and $z_i = 1$ if the ith person is in the knowledge group.

We assume each of these indicator variables is equally likely to be 0 or 1 a priori, so they have the prior $z_i \sim \text{Bernoulli}\big(1/2\big)$. For the guessing group, we assume that the rate is $\psi = 1/2$. For the knowledge group, we use a prior where all rate possibilities greater than $1/2$ are equally likely, so that $\phi \sim \text{Uniform}\big(0.5, 1\big)$.

This type of model is known as a *latent-mixture* model, because the data are assumed to be generated by two different processes that combine or mix, and important properties of that mixture are unobserved or latent. In this case, the two components that mix are the guessing and knowledge processes, and the group membership of each person is latent.

The script `Exams_1.txt` implements the graphical model in WinBUGS:

```
# Exam Scores
model{
  # Each Person Belongs To One Of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
  }
  # First Group Guesses
  psi <- 0.5
  # Second Group Has Some Unknown Greater Rate Of Success
  phi ~ dbeta(1,1)I(0.5,1)
  # Data Follow Binomial With Rate Given By Each Person's Group Assignment
  for (i in 1:p){
    theta[i] <- equals(z[i],0)*psi+equals(z[i],1)*phi
    k[i] ~ dbin(theta[i],n)
  }
}
```

The code `Exams_1.m` or `Exams_1.R` makes inferences about group membership, and the success rate of the knowledge group, using the model.

Exercises

Exercise 6.1.1 Draw some conclusions about the problem from the posterior distribution. Who belongs to what group, and how confident are you?

Exercise 6.1.2 The initial allocations of people to the two groups in this code is random, and so will be different every time you run it. Check that this does not affect the final results from sampling.

Exercise 6.1.3 Include an extra person in the exam, with a score of 28 out of 40. What does their posterior for z tell you? Now add four extra people, all with the score 28 out of 40. Explain the change these extra people make to the inference.

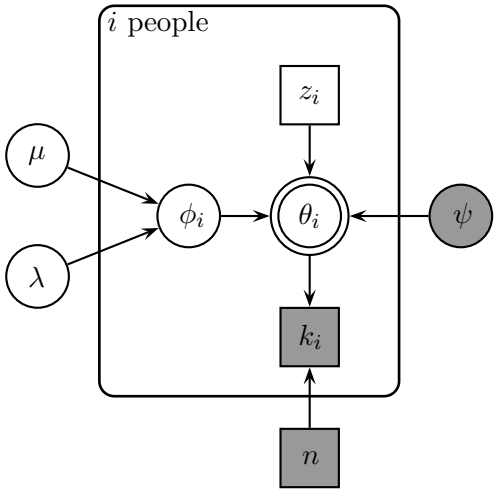
Exercise 6.1.4 What happens if you change the prior on the success rate of the second group to be uniform over the whole range from 0 to 1, and so allow for worse-than-guessing performance?

Exercise 6.1.5 What happens if you change the initial expectation that everybody is equally likely to belong to either group, and have an expectation that people generally are not guessing, with (say), $z_i \sim \text{Bernoulli}(0.9)$?

6.2 Exam scores with individual differences

The previous example shows how sampling can model data as coming from a mixture of sources, and infer properties of these latent groups. But the specific model has at least one big weakness, which is that it assumes all the people in the knowledge group have exactly the same rate of success on the questions.

One straightforward way to allow for individual differences in the knowledge group is to extend the model hierarchically. This involves drawing the success rate for each of the people in the knowledge group from an over-arching distribution. One convenient (but not perfect) choice for this "individual differences" distribution is a Gaussian. It is a natural statistical model for individual variation, at least in the absence of any richer theory. But it has the problem of allowing for success rates below zero and above one. An inelegant but practical and effective way to deal with this is simply to restrict the sampled success rates to the valid range.



$$
\begin{aligned}
z_i &\sim \text{Bernoulli}(0.5) \\
\mu &\sim \text{Uniform}(0.5, 1) \\
\lambda &\sim \text{Gamma}(0.001, 0.001) \\
\phi_i &\sim \text{Gaussian}(\mu, \lambda)_{\mathcal{I}(0,1)} \\
\psi &\leftarrow 0.5 \\
\theta_i &\leftarrow \begin{cases} \phi_i \text{ if } z_i = 1 \\ \psi \text{ if } z_i = 0 \end{cases} \\
k_i &\sim \text{Binomial}(\theta_i, n)
\end{aligned}
$$

Fig. 6.2 Graphical model for inferring membership of two latent groups, with different rates of success in answering exam questions, allowing for individual differences in the knowledge group.

A graphical model that implements this idea is shown in Figure 6.2. It extends the original model by having a knowledge group success rate ϕ_i for the ith person. These success rates are drawn from a Gaussian distribution with mean μ and precision

Box 6.1 Assessing and improving convergence

In a perfect world, a single MCMC chain would immediately begin drawing samples from the posterior distribution, and the only computational issue would be how many are needed to form a sufficiently precise approximation. This ideal state of affairs is often not what happens, and latent-mixture models are notorious for needing convergence checks. So, this is a good place to list some checks (see also Gelman, 1996; Gelman & Hill, 2007).

The basic principle is that, when the sampling process has converged, chains with substantially different starting values should be indistinguishable from each other. One implication of this requirement is that chains should vary around a constant mean, so a slow drift up or down signals a problem. And, if the sampling process has converged, each individual chain should look like a "fat hairy caterpillar," because this visual appearance is generated when successive values are relatively independent. As a formal test for convergence, the $\hat{R}$ statistic (Gelman & Rubin, 1992) is widely used. It is basically a measure of between-chain to within-chain variance, and so values close to 1 indicate convergence. As a rule of thumb, values higher than 1.1 are (deeply) suspect. If you were not paying much attention to the `rhat` values WinBUGS is returning to Matlab and R in previous modeling exercises, now is a good time to start checking them.

There are three basic remedies for a lack of convergence, easily implemented in WinBUGS for any model. The first is simply to collect many more samples, or more chains of samples, and wait (and hope) for convergence. The second is to increase the number of *burn-in* samples, which are initial samples in a chain that are discarded. This will be effective if separate chains are sensitive to their starting points, and take some time to converge. A worked example of this is presented in Section 11.2. The third is to *thin* the samples, by retaining only one out of every n. This will be effective if a chain is autocorrelated, with lack of independence between samples. A worked example of this is presented in Section 3.6. There are other, more advanced, methods for improving convergence in WinBUGS, involving changing the model itself. Worked examples of the *parameter expansion* method are presented in Sections 11.3 and 14.2.

λ. The mean μ is given a uniform prior between 0.5 and 1.0, consistent with the original assumption that people in the knowledge group have a greater-than-chance success rate.

Box 6.2 Scripts for graphical models

The scripts that implement graphical models in WinBUGS are declarative, rather than procedural. This means the order of the commands does not matter. All that a script does is define the observed and unobserved variables in a graphical model, saying how they are distributed, and how they relate to each other. This is inherently a structure, rather than a process, and so order is not important. In practice this means, for example, that a separate loop is not needed in a script like `Exam_2.txt` to define `k[i]`, `z[i]`, and `phi[i]`. Exactly the same graphical model would be defined if they were all placed inside one `for (i in 1:p)` loop. Sometimes, however, it is conceptually clearer to use separate loops to implement different parts of a graphical model.

The script `Exams_2.txt` implements the graphical model in WinBUGS:

```
# Exam Scores With Individual Differences
model{
  # Rates Given By Each Person's Group Assignment
  for (i in 1:p){
    theta[i] <- equals(z[i],0)*psi+equals(z[i],1)*phi[i]
    k[i] ~ dbin(theta[i],n)
  }
  # Each Person Belongs To One Of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
  }
  # The Second Group Allows Individual Differences
  for (i in 1:p){
    phi[i] ~ dnorm(mu,lambda)I(0,1)
  }
  # First Group Guesses
  psi <- 0.5
  # Second Group Mean, Precision (And Standard Deviation)
  mu ~ dbeta(1,1)I(.5,1) # >0.5 Average Success Rate
  lambda ~ dgamma(.001,.001)
  sigma <- 1/sqrt(lambda)
  # Posterior Predictive For Second Group
  predphi ~ dnorm(mu,lambda)I(0,1)
}
```

Notice that the code includes a variable `predphi` that draws success rates from the inferred Gaussian distribution of the knowledge group.

The code `Exams_2.m` or `Exams_2.R` makes inferences about group membership, the success rate of each person in the knowledge group, and the mean and standard deviation of the over-arching Gaussian for the knowledge group.

Exercises

Exercise 6.2.1 Compare the results of the hierarchical model with the original model that did not allow for individual differences.

Exercise 6.2.2 Interpret the posterior distribution of the variable `predphi`. How does this distribution relate to the posterior distribution for `mu`?

Exercise 6.2.3 In what sense could the latent assignment of people to groups in this case study be considered a form of model selection?

6.3 Twenty questions

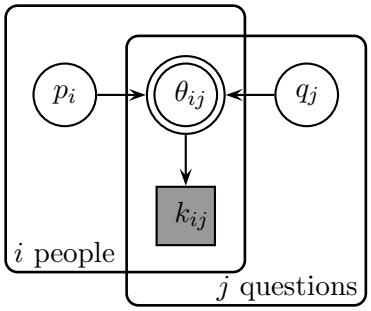
Suppose a group of 10 people attend a lecture, and are asked a set of 20 questions afterwards, with every answer being either correct or incorrect. The pattern of data is shown in Table 6.1. From this pattern of correct and incorrect answers we want to infer two things. The first is how well each person attended to the lecture. The second is how hard each of the questions was.

Table 6.1 Correct and incorrect answers for 10 people on 20 questions.

	Question																		
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Person 1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	0	0
Person 2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 3	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Person 4	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
Person 5	1	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	0	0
Person 6	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
Person 7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Person 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 9	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
Person 10	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0

One way to make these inferences is to specify a model of how a person's attentiveness and a question's difficulty combine to give an overall probability that the question will be answered correctly. A very simple model involves assuming that each person listens to some proportion of the lecture, and that each question has some probability of being answered correctly if the person was listening at the right point in the lecture.

A graphical model that implements this idea is shown in Figure 6.3. Under the model, if the ith person's probability of listening is p_i, and the jth question's probability of being answered correctly if the relevant information is heard is q_j,



$$p_i, q_j \sim \text{Beta}(1,1)$$
$$\theta_{ij} \leftarrow p_i q_j$$
$$k_{ij} \sim \text{Bernoulli}(\theta_{ij})$$

Fig. 6.3 Graphical model for inferring the rate people listened to a lecture, and the difficulty of the questions.

then the probability the ith person will answer the jth question correctly is just $\theta_{ij} = p_i q_j$. The observed pattern of correct and incorrect answers, where $k_{ij} = 1$ if the ith person answered the jth question correctly, and $k_{ij} = 0$ if they did not, then is a draw from a Bernoulli distribution with probability θ_{ij}.

The script `TwentyQuestions.txt` implements the graphical model in WinBUGS:

```
# Twenty Questions
model{
  # Correctness Of Each Answer Is Bernoulli Trial
  for (i in 1:np){
    for (j in 1:nq){
      k[i,j] ~ dbern(theta[i,j])
    }
  }
  # Probability Correct Is Product Of Question By Person Rates
  for (i in 1:np){
    for (j in 1:nq){
      theta[i,j] <- p[i]*q[j]
    }
  }
  # Priors For People and Questions
  for (i in 1:np){
    p[i] ~ dbeta(1,1)
  }
  for (j in 1:nq){
    q[j] ~ dbeta(1,1)
  }
}
```

The code `TwentyQuestions.m` or `TwentyQuestions.R` makes inferences about the data in Table 6.1 using the model.

Exercises

Exercise 6.3.1 Draw some conclusions about how well the various people listened, and about the difficulties of the various questions. Do the marginal posterior distributions you are basing your inference on seem intuitively reasonable?

Exercise 6.3.2 Now suppose that three of the answers were not recorded, for whatever reason. Our new data set, with missing data, now takes the form shown in Table 6.2. Bayesian inference will automatically make predictions about these missing values (i.e., "fill in the blanks") by using the same probabilistic model that generated the observed data. Missing data are entered as `nan` ("not a number") in Matlab, and `NA` ("not available") in R or WinBUGS. Including the variable `k` as one to monitor when sampling will then provide posterior values for the missing values. That is, it provides information about the relative likelihood of the missing values being each of the possible alternatives, using the statistical model and the available data. Look through the Matlab or R code to see how all of this is implemented in the second data set. Run the code, and interpret the posterior distributions for the three missing values. Are they reasonable inferences?

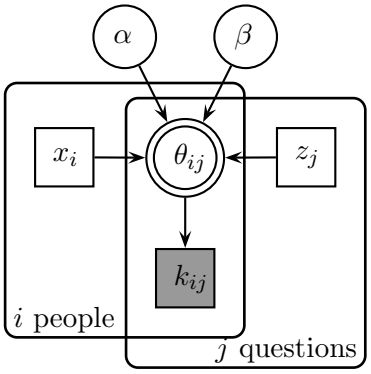
Table 6.2 Correct, incorrect, and missing answers for 10 people on 20 questions.

	Question																		
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Person 1	1	1	1	1	0	0	1	1	0	1	0	0	?	0	0	1	0	1	0	0
Person 2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 3	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0
Person 4	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
Person 5	1	0	1	1	0	1	1	1	0	1	0	0	1	0	0	0	0	1	0	0
Person 6	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
Person 7	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Person 8	0	0	0	0	?	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Person 9	0	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1
Person 10	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	?	0	0

Exercise 6.3.3 The definition of the accuracy for a person on a question in terms of the product $\theta_{ij} = p_i q_j$ is very simple to understand, but other models of the interaction between person ability and question difficulty are used in psychometric models. For example, the Rasch model (e.g., Andrich, 1988) uses $\theta_{ij} = \exp(p_i - q_j) / (1 + \exp(p_i - q_j))$. Change the graphical model to implement the Rasch model.

6.4 The two-country quiz

Suppose a group of people take a historical quiz, and each answer for each person is scored as correct or incorrect. Some of the people are Thai, and some are Moldovan.



$$
\begin{aligned}
\alpha &\sim \text{Uniform}(0, 1) \\
\beta &\sim \text{Uniform}(0, \alpha) \\
x_i &\sim \text{Bernoulli}(0.5) \\
z_j &\sim \text{Bernoulli}(0.5) \\
\theta_{ij} &\leftarrow \begin{cases} \alpha & \text{if } x_i = z_j \\ \beta & \text{if } x_i \neq z_j \end{cases} \\
k_{ij} &\sim \text{Bernoulli}(\theta_{ij})
\end{aligned}
$$

Fig. 6.4 Graphical model for inferring the country of origin for people and questions.

Some of the questions are about Thai history, and it is more likely the answer would be known by a Thai person than a Moldovan. The rest of the questions are about Moldovan history, and it is more likely the answer would be known by a Moldovan than a Thai.

We do not know who is Thai or Moldovan, and we do not know the content of the questions. All we have are the data shown in Table 6.3. Spend some time just looking at the data, and try to infer which people are from the same country, and which questions relate to their country.

Table 6.3 Correct and incorrect answers for 8 people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0

A good way to make these inferences formally is to assume there are two types of answers. For those where the nationality of the person matches the origin of the question, the answer will be correct with high probability. For those where a person is being asked about the other country, the answer will have a very low probability of being correct.

A graphical model that implements this idea is shown in Figure 6.4. The rate α is the (expected to be high) probability of a person from a country correctly

answering a question about their country's history. The rate β is the (expected to be low) probability of a person correctly answering a question about the other country's history. To capture the knowledge about the rates, the priors constrain $\alpha \geq \beta$, by defining `alpha` $\sim$ `dunif(0,1)` and `beta` $\sim$ `dunif(0,alpha)`. At first glance, this might seem inappropriate, since it specifies a prior for one parameter in terms of another (unknown, and being inferred) parameter. Conceptually, it is clearer to think of this syntax as a (perhaps clumsy) way to specify a *joint* prior over α and β in which the $\alpha \geq \beta$. Graphically, the parameter space over (α, β) is a unit square, and the prior being specified is the half of the square on one side of the diagonal line $\alpha = \beta$.

In the remainder of the graphical model, the binary indicator variable x_i assigns the ith person to one or other country, and z_j similarly assigns the jth question to one or other country. The probability the ith person will answer the jth question correctly is θ_{ij}, which is simply α if the country assignments match, and β if they do not. Finally, the actual data k_{ij} indicating whether or not the answer was correct follow a Bernoulli distribution with rate θ_{ij}.

The script `TwoCountryQuiz.txt` implements the graphical model in WinBUGS:

```
# The Two Country Quiz
model{
  # Probability of Answering Correctly
  alpha ~ dunif(0,1)    # Match
  beta ~ dunif(0,alpha) # Mismatch
  # Group Membership For People and Questions
  for (i in 1:nx){
    x[i] ~ dbern(0.5)
    x1[i] <- x[i]+1
  }
  for (j in 1:nz){
    z[j] ~ dbern(0.5)
    z1[j] <- z[j]+1
  }
  # Probability Correct For Each Person-Question Combination By Groups
  for (i in 1:nx){
    for (j in 1:nz){
      theta[i,j,1,1] <- alpha
      theta[i,j,1,2] <- beta
      theta[i,j,2,1] <- beta
      theta[i,j,2,2] <- alpha
    }
  }
  # Data Are Bernoulli By Rate
  for (i in 1:nx){
    for (j in 1:nz){
      k[i,j] ~ dbern(theta[i,j,x1[i],z1[j]])
    }
  }
}
```

The code `TwoCountryQuiz.m` or `TwoCountryQuiz.R` makes inferences about the data in Table 6.3 using the model.

Exercises

Exercise 6.4.1 Interpret the posterior distributions for `x[i]`, `z[j]`, `alpha`, and `beta`. Do the formal inferences agree with your original intuitions?

Exercise 6.4.2 The priors on the probabilities of answering correctly capture knowledge about what it means to match and mismatch, by imposing an order constraint $\alpha \geq \beta$. Change the code so that this information is not included, by using priors `alpha`∼`dbeta(1,1)` and `beta`∼`dbeta(1,1)`. Run a few chains against the same data, until you get an inappropriate, and perhaps counter-intuitive, result. The problem that is being encountered is known as model indeterminacy or label-switching. Describe the problem, and discuss why it comes about.

Exercise 6.4.3 Now suppose that three extra people enter the room late, and begin to take the quiz. One of them (Late Person 1) has answered the first four questions, the next (Late Person 2) has only answered the first question, and the final new person (Late Person 3) is still sharpening their pencil, and has not started the quiz. This situation can be represented as an updated data set, now with missing data, as in Table 6.4. Interpret the inferences the model makes about the nationality of the late people, and whether or not they will get the unfinished questions correct.

Table 6.4 Correct, incorrect, and missing answers for 8 people and 3 late people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0
Late Person 1	1	0	0	1	?	?	?	?
Late Person 2	0	?	?	?	?	?	?	?
Late Person 3	?	?	?	?	?	?	?	?

Exercise 6.4.4 Finally, suppose that you are now given the correctness scores for a set of 10 new people, whose data were not previously available, but who form part of the same group of people we are studying. The updated data set is shown in Table 6.5. Interpret the inferences the model makes about the nationality of the new people. Revisit the inferences about the late people, and whether or not they will get the unfinished questions correct. Does the

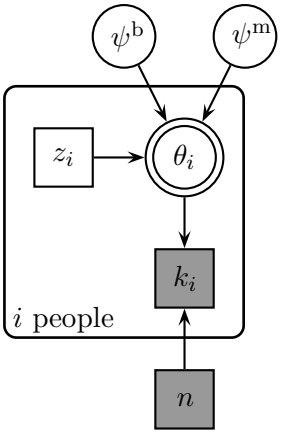
inference drawn by the model for the third late person match your intuition? There is a problem here. How could it be fixed?

Table 6.5 Correct, incorrect, and missing answers for 8 people, 3 late people, and 10 new people on 8 questions.

	Question							
	A	B	C	D	E	F	G	H
New Person 1	1	0	0	1	1	0	0	1
New Person 2	1	0	0	1	1	0	0	1
New Person 3	1	0	0	1	1	0	0	1
New Person 4	1	0	0	1	1	0	0	1
New Person 5	1	0	0	1	1	0	0	1
New Person 6	1	0	0	1	1	0	0	1
New Person 7	1	0	0	1	1	0	0	1
New Person 8	1	0	0	1	1	0	0	1
New Person 9	1	0	0	1	1	0	0	1
New Person 10	1	0	0	1	1	0	0	1
Person 1	1	0	0	1	1	0	0	1
Person 2	1	0	0	1	1	0	0	1
Person 3	0	1	1	0	0	1	0	0
Person 4	0	1	1	0	0	1	1	0
Person 5	1	0	0	1	1	0	0	1
Person 6	0	0	0	1	1	0	0	1
Person 7	0	1	0	0	0	1	1	0
Person 8	0	1	1	1	0	1	1	0
Late Person 1	1	0	0	1	**?**	**?**	**?**	**?**
Late Person 2	0	**?**	**?**	**?**	**?**	**?**	**?**	**?**
Late Person 3	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

6.5 Assessment of malingering

Armed with the knowledge from the previous sections, we now consider the practical challenge of detecting if people cheat on a test. For example, people who have been in a car accident may seek financial compensation from insurance companies by feigning cognitive impairment such as pronounced memory loss. When these people are confronted with a memory test that is intended to measure the extent of their impairment, they may deliberately under-perform. This behavior is called malingering, and it may be accompanied by performance much worse than that displayed by real amnesiacs. Sometimes, for example, malingerers may perform substantially below chance.



$$
\begin{aligned}
\psi^{\mathrm{b}} &\sim \mathrm{Uniform}(0.5, 1) \\
\psi^{\mathrm{m}} &\sim \mathrm{Uniform}(0, \psi^{\mathrm{b}}) \\
z_i &\sim \mathrm{Bernoulli}(0.5) \\
\theta_i &\leftarrow \begin{cases} \psi^{\mathrm{b}} & \text{if } z_i = 0 \\ \psi^{\mathrm{m}} & \text{if } z_i = 1 \end{cases} \\
k_i &\sim \mathrm{Binomial}(\theta_i, n)
\end{aligned}
$$

Fig. 6.5 Graphical model for the detection of malingering.

Malingering is not, however, always easy to detect, but is naturally addressed by latent-mixture modeling. Using this approach, it is possible to infer which of two categories—those who malinger, and those who are truthful or *bona fide*—each person belongs to, and quantify the confidence in each of these classifications.

We consider an experimental study on malingering, in which each of $p = 22$ participants completed a memory test (Ortega, Wagenmakers, Lee, Markowitsch, & Piefke, 2012). One group of participants was told to do their best. These are the *bona fide* participants. The other group of participants was told to under-perform by deliberately simulating amnesia. These are the malingerers. Out of a total of $n = 45$ test items, the participants get 45, 45, 44, 45, 44, 45, 45, 45, 45, 45, 30, 20, 6, 44, 44, 27, 25, 17, 14, 27, 35, and 30 correct. Because this was an experimental study, we know that the first 10 participants were *bona fide* and the next 12 were instructed to malinger.

The first analysis is straightforward, and uses the graphical model shown in Figure 6.5. We assume that all *bona fide* participants have the same ability, and so have the same rate ψ_{b} of answering each question correctly. For the malingerers, the rate of answering questions correctly is given by ψ_{m}, and $\psi_{\mathrm{b}} > \psi_{\mathrm{m}}$.

The script `Malingering_1.txt` implements the graphical model in WinBUGS:

```
# Malingering
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(0.5)
    z1[i] <- z[i]+1
  }
  # Bona Fide Group has Unknown Success Rate Above Chance
  psi[1] ~ dunif(0.5,1)
  # Malingering Group has Unknown Success Rate Below Bona Fide
  psi[2] ~ dunif(0,psi[1])
  # Data are Binomial with Group Rate for Each Person
  for (i in 1:p){
```

```
    theta[i] <- psi[z1[i]]
    k[i] ~ dbin(theta[i],n)
  }
}
```

Notice the restriction in the `dunif` definition of `psi[2]`, which prevents the indeterminacy or label-switching problem by ensuring that $\psi_b > \psi_m$.

The code `Malingering_1.m` or `Malingering_1.R` applies the model to the data.

Exercise

Exercise 6.5.1 What are your conclusions about group membership? Did all of the participants follow the instructions?

6.6 Individual differences in malingering

As before, it may seem restrictive to assume that all members of a group have the same chance of answering correctly. So, now we assume that the ith participant in each group has a unique rate of answering questions correctly, θ_i, which is constrained by group-level distributions. In Section 6.2, we used group-level Gaussians. The problem with that approach is that values can lie outside the range 0 to 1. These values were just censored in Section 6.2, but this is not quite technically correct, and is certainly not elegant.[1]

One of several alternatives is to assume that instead of being Gaussian, the group-level distribution is Beta(α, β). Because the Beta distribution is defined on the interval from 0 to 1 it respects the natural boundaries of rates. So we now have a model in which each individual binomial rate parameter is constrained by a group-level beta distribution. This complete model is known as the beta-binomial (e.g., Merkle, Smithson, & Verkuilen, 2011; J. B. Smith & Batchelder, 2010).

It is useful to transform the α and β parameters from the beta distribution to a group mean $\mu = \alpha/(\alpha + \beta)$ and a measure $\lambda = \alpha + \beta$ that can be conceived of as a precision, in the sense that as it increases the variability of the distribution decreases. It is then straightforward to assign uniform priors to both μ_b, the group-level mean for the *bona fide* participants, and μ_m, the group-level mean for the malingerers. This assignment does not, however, reflect our knowledge that $\mu_b > \mu_m$. To capture this knowledge, we could define `dunif(0,mubon)`, as done in the previous model.

However, for this model we apply a different approach. We first define μ_m as the additive combination of μ_b and a difference parameter, so that $\text{logit}(\mu_m) = \text{logit}(\mu_b) - \mu_d$. Note that this is an additive combination on the logit scale,

[1] WinBUGS conceptually conflates censoring and truncation in the `I(,)` notation, which is the cause of the technical problem. JAGS has the advantage of dealing with these two related concepts coherently.

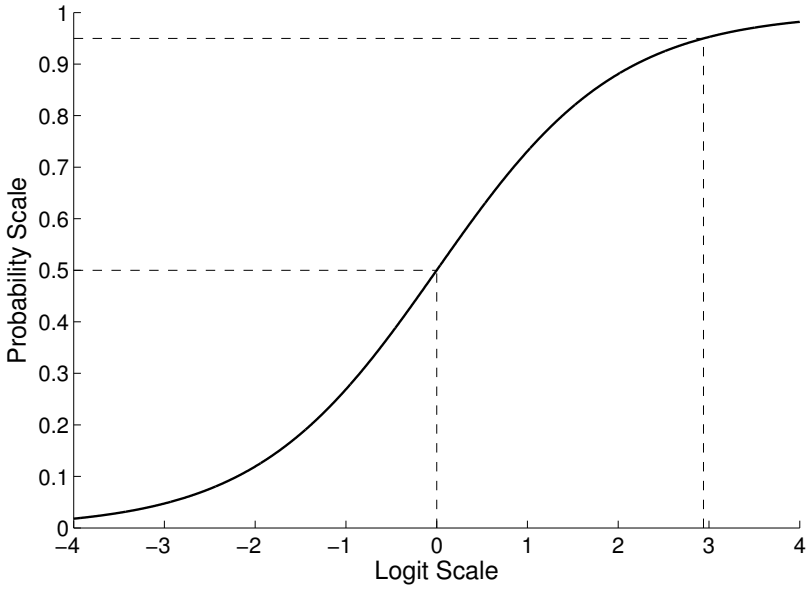


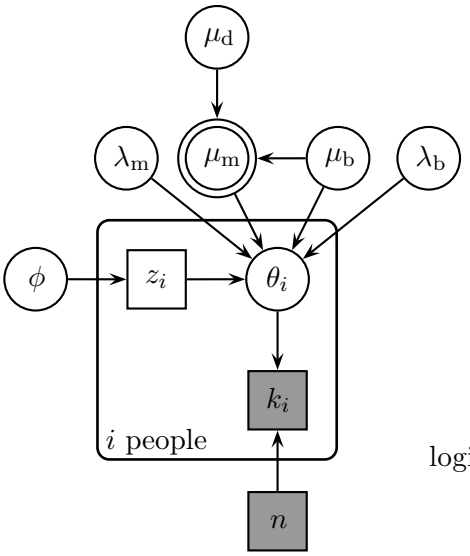
Fig. 6.6 The logit transformation. Probabilities range from 0 to 1 and are mapped to the entire set of real numbers using the logit transform.

as is customary in beta-binomial models. The logit transformation is defined as $\text{logit}(\theta) \equiv \ln(\theta/(1-\theta))$ and it transforms values on the rate scale, ranging from 0 to 1, to values on the logit scale, ranging from $-\infty$ to ∞. The logit transformation is shown in Figure 6.6, including two specific examples with the logit value 0 corresponding to probability 0.5, and the logit probability 2.94 corresponding to probability 0.95.

The prior for $\mu_{\text{d}} \sim \text{Gaussian}\big(0, 0.5\big)_{\mathcal{I}(0,\infty)}$ is a positive-only Gaussian distribution. This ensures that the group mean of the *bona fide* participants is always larger than that of the malingerers. Finally, note that the base rate of malingering ϕ, which was previously fixed to 0.5, is now assigned a relatively wide beta prior distribution that is centered around 0.5. This means the model uses the data to infer group membership and at the same time learn about the base rate.

A graphical model that implements the above ideas is shown in Figure 6.7. The script `Malingering_2.txt` implements the graphical model in WinBUGS:

```
# Malingering, with Individual Differences
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
    z[i] ~ dbern(phi) # phi is the Base Rate
    z1[i] <- z[i]+1
  }
  # Relatively Uninformative Prior on Base Rate
  phi ~ dbeta(5,5)
  # Data are Binomial with Rate Given by
  # Each Persons Group Assignment
```



$$
\begin{aligned}
\mu_{\mathrm{b}} &\sim \mathrm{Beta}(1,1) \\
\mu_{\mathrm{d}} &\sim \mathrm{Gaussian}(0,0.5)_{\mathcal{I}(0,\infty)} \\
\lambda_{\mathrm{b}} &\sim \mathrm{Uniform}(40,800) \\
\lambda_{\mathrm{m}} &\sim \mathrm{Uniform}(4,100) \\
z_i &\sim \mathrm{Bernoulli}(\phi) \\
\theta_i &\sim \begin{cases} \mathrm{Beta}(\mu_{\mathrm{b}}\lambda_{\mathrm{b}}, (1-\mu_{\mathrm{b}})\,\lambda_{\mathrm{b}}) & \text{if } z_i = 0 \\ \mathrm{Beta}(\mu_{\mathrm{m}}\lambda_{\mathrm{m}}, (1-\mu_{\mathrm{m}})\,\lambda_{\mathrm{m}}) & \text{if } z_i = 1 \end{cases} \\
k_i &\sim \mathrm{Binomial}(\theta_i, n) \\
\mathrm{logit}\mu_{\mathrm{m}} &\leftarrow \mathrm{logit}\mu_{\mathrm{b}} - \mu_{\mathrm{d}} \\
\phi &\sim \mathrm{Beta}(5,5)
\end{aligned}
$$

Fig. 6.7 Graphical model for inferring membership of two latent groups, consisting of malingerers and *bona fide* participants.

```
  for (i in 1:p){
    k[i] ~ dbin(theta[i,z1[i]],n)
    theta[i,1] ~ dbeta(alpha[1],beta[1])
    theta[i,2] ~ dbeta(alpha[2],beta[2])
  }
  # Transformation to Group Mean and Precision
  alpha[1] <- mubon * lambdabon
  beta[1] <- lambdabon * (1-mubon)
  # Additivity on Logit Scale
  logit(mumal) <- logit(mubon) - mudiff
  alpha[2] <- mumal * lambdamal
  beta[2]  <- lambdamal * (1-mumal)
  # Priors
  mubon ~ dbeta(1,1)
  mudiff ~ dnorm(0,0.5)I(0,) # Constrained to be Positive
  lambdabon ~ dunif(40,800)
  lambdamal ~ dunif(4,100)
}
```

The code `Malingering_2.m` or `Malingering_2.R` allows you to draw conclusions about group membership and the success rate of the two groups.

Exercises

Exercise 6.6.1 Is the inferred rate of malingering consistent with what is known about the instructions given to participants?

Exercise 6.6.2 Assume you know that the base rate of malingering is 10%. Change the WinBUGS script to reflect this knowledge. Do you expect any differences?

Exercise 6.6.3 Assume you know for certain that participants 1, 2, and 3 are *bona fide*. Change the code to reflect this knowledge.

Exercise 6.6.4 Suppose you add a new participant. What number of questions answered correctly by this participant would lead to the greatest uncertainty about their group membership?

Exercise 6.6.5 Try to solve the label-switching problem by using the `dunif(0,mubon)` approach instead of the logit transform.

Exercise 6.6.6 Why are the priors for λ_b and λ_m different?

6.7 Alzheimer's recall test cheating

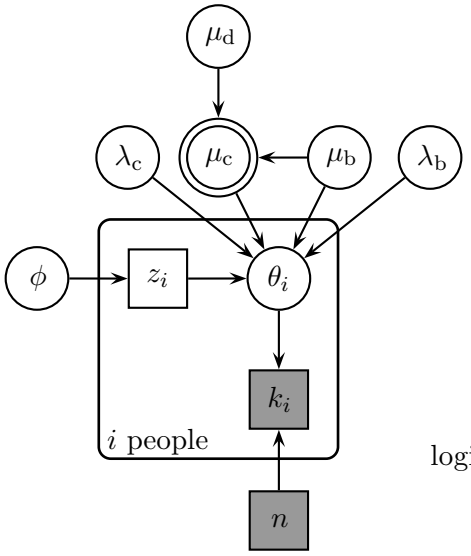
In this section, we apply the same latent-mixture model shown in Figure 6.7 to different memory test data. Simple recognition and recall tasks are an important part of screening for Alzheimer's Disease and Related Disorders (ADRD), and are sometimes administered over the telephone. This practice raises the possibility of people cheating by, for example, writing down the words they are being asked to remember.

The data we use come from an informal experiment, in which 118 people were either asked to complete the test normally, or instructed to cheat. The particular test used was a complicated sequence of immediate and delayed free recall tasks, which we simplify to give a simple score correct out of 40 for each person. By design, there are 61 *bona fide* people who are known to have done the task as intended, and 57 people who are known to have cheated.

This graphical model is shown in Figure 6.8, and is essentially the same as for the previous example on malingering in Figure 6.7. It changes the names of variables from malingering to cheating as appropriate, uses different priors on the precisions of the group distributions, and makes the mean of accuracy rate for the cheaters *higher* than that of the *bona fide* people, since the impact of cheating is to recall more words than would otherwise be the case.

The script `Cheating.txt` implements the analysis in WinBUGS:

```
# Cheating Latent-Mixture Model
model{
  # Each Person Belongs to One of Two Latent Groups
  for (i in 1:p){
     z[i] ~ dbern(phi) # phi is the Base Rate
     z1[i] <- z[i]+1
  }
  # Relatively Uninformative Prior on Base Rate
  phi ~ dbeta(5,5)
  # Data are Binomial with Rate Given by
  # Each Persons Group Assignment
  for (i in 1:p){
```



$$
\begin{aligned}
\mu_{\mathrm{b}} &\sim \mathrm{Beta}(1,1) \\
\mu_{\mathrm{d}} &\sim \mathrm{Gaussian}(0,0.5)_{\mathcal{I}(0,\infty)} \\
\lambda_{\mathrm{b}} &\sim \mathrm{Uniform}(5,50) \\
\lambda_{\mathrm{c}} &\sim \mathrm{Uniform}(5,50) \\
z_i &\sim \mathrm{Bernoulli}(\phi) \\
\theta_i &\sim \begin{cases} \mathrm{Beta}(\mu_{\mathrm{b}}\lambda_{\mathrm{b}}, (1-\mu_{\mathrm{b}})\,\lambda_{\mathrm{b}}) & \text{if } z_i = 0 \\ \mathrm{Beta}(\mu_{\mathrm{c}}\lambda_{\mathrm{c}}, (1-\mu_{\mathrm{c}})\,\lambda_{\mathrm{c}}) & \text{if } z_i = 1 \end{cases} \\
k_i &\sim \mathrm{Binomial}(\theta_i, n) \\
\mathrm{logit}\mu_{\mathrm{c}} &\leftarrow \mathrm{logit}\mu_{\mathrm{b}} + \mu_{\mathrm{d}} \\
\phi &\sim \mathrm{Beta}(5,5)
\end{aligned}
$$

Fig. 6.8 Graphical model for inferring membership of two latent groups, consisting of cheaters and *bona fide* people in a memory test.

```
    k[i] ~ dbin(theta[i,z1[i]],n)
    thetatmp[i,1] ~ dbeta(alpha[1],beta[1])
    theta[i,1] <- max(.01,min(.99,thetatmp[i,1]))
    thetatmp[i,2] ~ dbeta(alpha[2],beta[2])
    theta[i,2] <- max(.01,min(.99,thetatmp[i,2]))
  }
  # Transformation to Group Mean and Precision
  alpha[1] <- mubon * lambdabon
  beta[1] <- lambdabon * (1-mubon)
  # Additivity on Logit Scale
  logit(muche) <- logit(mubon) + mudiff # Note the "+"
  alpha[2] <- muche * lambdache
  beta[2] <- lambdache * (1-muche)
  # Priors
  mubon ~ dbeta(1,1)
  mudiff ~ dnorm(0,0.5)I(0,) # Constrained to be Positive
  lambdabon ~ dunif(5,40)
  lambdache ~ dunif(5,40)
  # Correct Count
  for (i in 1:p){
    pct[i] <- equals(z[i],truth[i])
  }
  pc <- sum(pct[1:p])
}
```

Note that the script includes a variable `pc` that keeps track of the accuracy of each classification made in sampling by comparing each person's latent assignment to the known truth from the experimental design.

The code `Cheating.m` or `Cheating.R` applies the graphical model to the data. We focus our analysis of the results firstly on the classification accuracy of the

Box 6.3 Undefined real result

In WinBUGS, error messages are called traps, and some traps are more serious than others. One of the more serious regularly occurring traps is "undefined real result." This trap indicates numerical overflow or underflow caused by a sample with very low likelihood. This can happen when you have not specified your model well enough. In particular, the prior distribution for a standard deviation may be too wide, thus allowing extreme values that are highly unlikely in light of the data (and, most often, also unlikely in light of prior knowledge). Initial values may also be to blame, and this is why it can be better to specify those yourself instead of having WinBUGS pick them automatically. Although the "undefined real result" trap can be a nuisance, in the end it may actually help you improve your model.

model. The top panel of Figure 6.9 summarizes the data, showing the distribution of correctly recalled words in both the *bona fide* and cheater groups. It is clear that cheaters generally recall more words, but that there is overlap between the groups.

One way to provide a benchmark classification accuracy is to consider the best possible cut-off. This is a total correct score below which a person is classified as *bona fide*, and at or above which they are classified as a cheater. The line in the bottom panel in Figure 6.9 shows the classification accuracy for all possible cut-offs, which peaks at 86.4% accuracy using the cut-off of 35. The gray distribution at the left of the panel is the posterior distribution of the `pc` variable, showing the range of accuracy achieved by the latent-mixture model.

Using a generative model to solve classification problems is unlikely to work as well as the best discriminative methods from machine learning and statistics. This is not because of failings of the Bayesian approach, but because the models we develop are imperfect accounts of how data are generated. If the focus is purely on prediction, other statistical approaches, including especially ones that combine the best aspects of generative and discriminative modeling, may be superior (e.g., Lasserre, Bishop, & Minka, 2006).

The advantage of the generative model is in providing details about the underlying processes assumed to produce the data, particularly by quantifying uncertainty. A good example of this important feature is shown in Figure 6.10, which shows the relationship between the total correct raw data score, and the posterior uncertainty about classification as a cheater, for each person. The broken lines connecting 35 people and a classification probability of 0.5 shows that the model infers people with scores above 35 as more likely than not to be cheaters. But it also shows how certain the model is about each classification, which provides more information (and more probabilistically coherent information) than many machine-learning methods.

This information about uncertainty is useful, for example, if there are costs or utilities associated with different classification decisions. Suppose that raising a

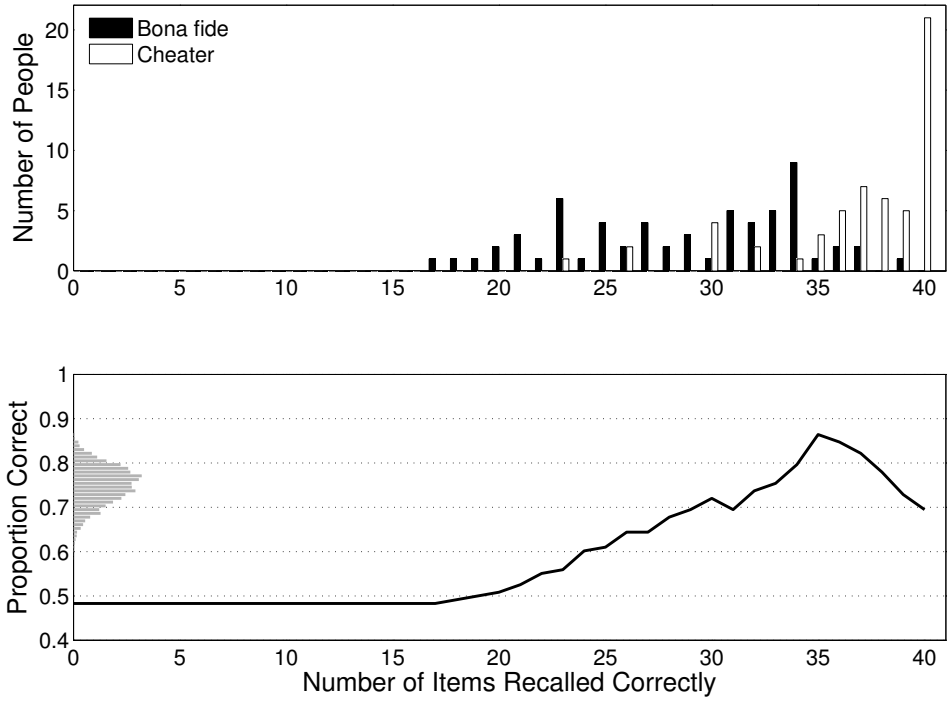


Fig. 6.9 The distribution of total correct recall scores for the Alzheimer's data, and classification performance. The top panel shows the distribution of scores for the *bona fide* and cheater groups. The bottom panel shows, with the line, the accuracy achieved using various cut-offs to separate the groups, and, with the distribution, the accuracy achieved by the latent-mixture model.

false-alarm and suspecting someone of cheating on the screening test costs \$25, perhaps through a wasted follow-up-in-person test, but that missing someone who cheated on the screening test costs \$100, perhaps through providing insurance that should have been withheld. With these utilities, the decision should be to classify people as *bona fide* only if it is four times more likely than them being a cheater. In other words, we need 80% certainty they are *bona fide*. The posterior distribution of the latent assignment variable $\mathbf{z}$ provides exactly this information. Under this set of utilities, as shown by the broken lines connecting 30 people and a classification probability of 0.2 in Figure 6.10, only people with a total correct score below 30 (not 35) should be treated as *bona fide*.

Exercises

Exercise 6.7.1 Suppose the utilities are very different, so that a false alarm costs \$100, because of the risk of litigation in a false accusation, but misses are relatively harmless, costing \$10 in wasted administrative costs. What decisions should be made about *bona fide* and cheating people now?

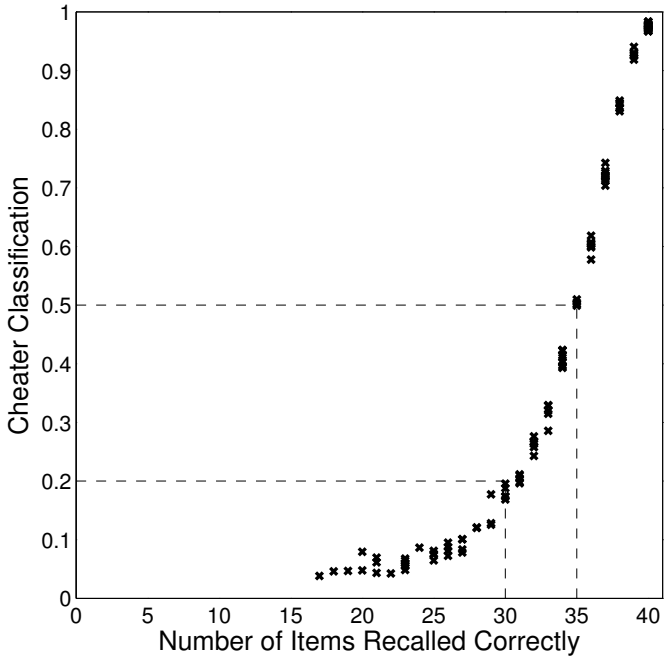


Fig. 6.10 The relationship between the number of items recalled correctly, and the posterior classification as belonging to the cheater group. Each cross corresponds to a person.

Exercise 6.7.2 What other potential information, besides the uncertainty about classification, does the model provide? Give at least one concrete example.

References

Agresti, A. (1992). Modelling patterns of agreement and disagreement. *Statistical Methods in Medical Research*, *1*, 201–218.

Ahn, W. Y., Busemeyer, J. R., Wagenmakers, E.-J., & Stout, J. C. (2008). Comparison of decision learning models using the generalization criterion method. *Cognitive Science*, *32*, 1376–1402.

Albers, W. (2001). Prominence theory as a tool to model boundedly rational decisions. In G. Gigerenzer & R. Selten (Eds.), *Bounded Rtionality: The Adaptive Toolbox* (pp. 297–317). Cambridge, MA: MIT Press.

Andrich, D. (1988). *Rasch Models for Measurement*. London: Sage.

Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine Learning*, *50*, 5–43.

Angus, J. E. (1994). The probability integral transform and related results. *SIAM Review*, *36*, 652–654.

Anscombe, F. J. (1963). Sequential medical trials. *Journal of the American Statistical Association*, *58*, 365–383.

Banerjee, M., Capozzoli, M., McSweeney, L., & Sinha, D. (1999). Beyond kappa: A review of interrater agreement measures. *The Canadian Journal of Statistics*, *27*, 3–23.

Bartlett, M. S. (1957). A comment on D. V. Lindley's statistical paradox. *Biometrika*, *44*, 533–534.

Basu, S., Banerjee, M., & Sen, A. (2000). Bayesian inference for kappa from single and multiple studies. *Biometrics*, *56*, 577–582.

Batchelder, W. H., & Riefer, D. M. (1980). Separation of storage and retrieval factors in free recall of clusterable pairs. *Psychological Review*, *87*, 375–397.

Batchelder, W. H., & Riefer, D. M. (1986). The statistical analysis of a model for storage and retrieval processes in human memory. *British Journal of Mathematical and Statistical Psychology*, *39*, 120–149.

Behseta, S., Berdyyeva, T., Olson, C. R., & Kass, R. E. (2009). Bayesian correction for attentuation of correlation in multi–trial spike count data. *Journal of Neurophysiology*, *101*, 2186–2193.

Bem, D. J. (2011). Feeling the future: Experimental evidence for anomalous retroactive influences on cognition and affect. *Journal of Personality and Social Psychology*, *100*, 407–425.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (2nd edn.). New York: Springer.

Berger, J. O., & Berry, D. A. (1988). The relevance of stopping rules in statistical inference. In S. S. Gupta & J. O. Berger (Eds.), *Statistical Decision Theory and Related Topics, Vol. 1* (pp. 29–72). New York: Springer Verlag.

Berger, J. O., & Delampady, M. (1987). Testing precise hypotheses. *Statistical Science*, *2*, 317–352.

Berger, J. O., & Mortera, J. (1999). Default Bayes factors for nonnested hypothesis testing. *Journal of the American Statistical Association*, *94*, 542–554.

Berger, J. O., & Pericchi, L. R. (1996). The intrinsic Bayes factor for model selection and prediction. *Journal of the American Statistical Association*, *91*, 109–122.

Berger, J. O., & Wolpert, R. L. (1988). *The Likelihood Principle* (2nd edn.). Hayward, CA: Institute of Mathematical Statistics.

Bergert, F. B., & Nosofsky, R. M. (2007). A response-time approach to comparing generalized rational and take-the-best models of decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *33*, 107–129.

Besag, J. (1989). A candidate's formula: A curious result in Bayesian prediction. *Biometrika*, *76*, 183.

Bowers, J. S., Vigliocco, G., & Haan, R. (1998). Orthographic, phonological, and articulatory contributions to masked letter and word priming. *Journal of Experimental Psychology: Human Perception and Performance*, *24*, 1705–1719.

Broemeling, L. D. (2009). *Bayesian Methods for Measures of Agreement*. Boca Raton, FL: CRC Press.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*, 434–455.

Brown, G. D. A., Neath, I., & Chater, N. (2007). A temporal ratio model of memory. *Psychological Review*, *114*, 539–576.

Brückner, H., & Bearman, P. (2005). After the promise: The STD consequences of adolescent virginity pledges. *Journal of Adolescent Health*, *36*, 271–278.

Burian, S. E., Liguori, A., & Robinson, J. H. (2002). Effects of alcohol on risk-taking during simulated driving. *Human Psychopharmacology*, *17*, 141–150.

Carey, S. (2001). Cognitive foundations of arithmetic: Evolutionary and ontogenetic. *Mind and Language*, *16*, 37–55.

Carey, S., & Sarnecka, B. W. (2006). The development of human conceptual representations. In M. Johnson & Y. Munakata (Eds.), *Processes of Change in Brain and Cognitive Development: Attention and Performance XXI* (pp. 473–496). Oxford: Oxford University Press.

Carlin, B. P., & Chib, S. (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, *57*, 473–484.

Chater, N., & Oaksford, M. (2000). The rational analysis of mind and behavior. *Synthese*, *122*, 93–131.

Chater, N., Tenenbaum, J. B., & Yuille, A. (2006). Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, *10*, 287–291.

Chechile, R. A. (1973). *The Relative Storage and Retrieval Losses in Short–Term Memory as a Function of the Similarity and Amount of Information Processing in the Interpolated Task*. Unpublished doctoral dissertation, University of Pittsburgh.

Chechile, R. A., & Meyer, D. L. (1976). A Bayesian procedure for separately estimating storage and retrieval components of forgetting. *Journal of Mathematical Psychology*, *13*, 269–295.

Chen, M.-H. (2005). Computing marginal likelihoods from a single MCMC output. *Statistica Neerlandica*, *59*, 16–29.

Chib, S. (1995). Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, *90*, 1313–1321.

Chib, S., & Jeliazkov, I. (2001). Marginal likelihood from the Metropolis–Hastings output. *Journal of the American Statistical Association*, *96*, 270–281.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*, 37–46.

Dawid, A. P. (2000). Comment on "The philosophy of statistics" by D. V. Lindley. *The Statistician*, *49*, 325–326.

de Finetti, B. (1974). *Theory of Probability, Vol. 1 and 2*. New York: John Wiley.

Dennis, S. J., & Humphreys, M. S. (2001). A context noise model of episodic word recognition. *Psychological Review*, *108*, 452–477.

Dennis, S. J., Lee, M. D., & Kinnell, A. (2008). Bayesian analysis of recognition memory: The case of the list-length effect. *Journal of Memory and Language*, *59*, 361–376.

Dickey, J. M. (1971). The weighted likelihood ratio, linear hypotheses on normal location parameters. *The Annals of Mathematical Statistics*, *42*, 204–223.

Dickey, J. M., & Lientz, B. P. (1970). The weighted likelihood ratio, sharp hypotheses about chances, the order of a Markov chain. *The Annals of Mathematical Statistics*, *41*, 214–226.

Dienes, Z. (2011). Bayesian versus orthodox statistics: Which side are you on? *Perspectives on Psychological Science*, *6*, 274–290.

Donner, A., & Wells, G. (1986). A comparison of confidence interval methods for the intraclass correlation coefficient. *Biometrics*, *42*, 401–412.

Dunwoody, P. T. (2009). Introduction to the special issue: Coherence and correspondence in judgment and decision making. *Judgment and Decision Making*, *4*, 113–115.

Edwards, W., Lindman, H., & Savage, L. J. (1963). Bayesian statistical inference for psychological research. *Psychological Review*, *70*, 193–242.

Ernst, M. O. (2005). A Bayesian view on multimodal cue integration. In G. Knoblich, I. M. Thornton, J. Grosjean, & M. Shiffrar (Eds.), *Human Body Perception From the Inside Out* (pp. 105–131). New York: Oxford University Press.

Estes, W. K. (2002). Traps in the route to models of memory and decision. *Psychonomic Bulletin & Review*, *9*, 3–25.

Evans, A. N., & Rooney, B. J. (2011). *Methods in Psychological Research* (2nd edn.). London: Sage.

Fleiss, J. L., Levin, B., & Paik, M. C. (2003). *Statistical Methods for Rates and Proportions* (3rd edn.). New York: John Wiley.

Forster, K. I., Mohan, K., & Hector, J. (2003). The mechanics of masked priming. In S. Kinoshita & S. J. Lupker (Eds.), *Masked Priming: The State of the Art* (pp. 3–38). New York, NY: Psychology Press.

Frye, D., Braisby, N., Lowe, J., Maroudas, C., & Nicholls, J. (1989). Young children's understanding of counting and cardinality. *Child Development*, *60*, 1158–1171.

Fuson, K. C. (1988). *Children's Counting and Concepts of Number*. New York: Springer-Verlag.

Gallistel, C. R. (2009). The importance of proving the null. *Psychological Review*, *116*, 439–453.

Gamerman, D., & Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Boca Raton, FL: Chapman & Hall/CRC.

Gelman, A. (1996). Inference and monitoring convergence. In W. R. Gilks, S. Richardson, & D. J. Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice* (pp. 131–143). Boca Raton (FL): Chapman & Hall/CRC.

Gelman, A. (2004). Parameterization and Bayesian modeling. *Journal of the American Statistical Association*, *99*, 537–545.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, *1*, 515–534.

Gelman, A., & Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge: Cambridge University Press.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science*, *7*, 457–472.

Geurts, H. M., Verté, S., Oosterlaan, J., Roeyers, H., & Sergeant, J. A. (2004). How specific are executive functioning deficits in attention deficit hyperactivity disorder and autism? *Journal of Child Psychology and Psychiatry*, *45*, 836–854.

Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic decision making. *Annual Review of Psychology*, *62*, 451-482.

Gigerenzer, G., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, *103*, 650–669.

Gigerenzer, G., & Todd, P. M. (1999). *Simple Heuristics That Make Us Smart*. New York: Oxford University Press.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice*. Boca Raton, FL: Chapman & Hall/CRC.

Gill, J. (2002). *Bayesian Methods: A Social and Behavioral Sciences Approach*. Boca Raton, FL: CRC Press.

Gönen, M., Johnson, W. O., Lu, Y., & Westfall, P. H. (2005). The Bayesian two-sample t test. *The American Statistician*, *59*, 252–257.

Grant, J. A. (1974). Evaluation of a screening program. *American Journal of Public Health*, *64*, 66–71.

Green, D. M., & Swets, J. A. (1966). *Signal detection theory and psychophysics*. New York: John Wiley.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, *82*, 711–732.

Griffiths, T. L., Kemp, C., & Tenenbaum, J. B. (2008). Bayesian models of cognition. In R. Sun (Ed.), *Cambridge Handbook of Computational Cognitive Modeling* (pp. 59–100). Cambridge, MA: Cambridge University Press.

Heit, E. (2000). Properties of inductive reasoning. *Psychonomic Bulletin & Review*, *7*, 569–592.

Heit, E., & Rotello, C. (2005). Are there two kinds of reasoning? In B. G. Bara, L. W. Barsalou, & M. Bucciarelli (Eds.), *Proceedings of the 27th Annual Conference of the Cognitive Science Society* (pp. 923–928). Mahwah, NJ: Erlbaum.

Hoijtink, H., Klugkist, I., & Boelen, P. (2008). *Bayesian Evaluation of Informative Hypotheses*. New York: Springer.

Hu, X., & Batchelder, W. H. (1994). The statistical analysis of general processing tree models with the EM algorithm. *Psychometrika*, *59*, 21–47.

Hyman, R. (1985). The Ganzfeld psi experiment: A critical appraisal. *Journal of Parapsychology*, *49*, 3–49.

Ivry, R. B. (1996). The representation of temporal information in perception and motor control. *Current Opinion in Neurobiology*, *14*, 225–232.

Jaynes, E. T. (1976). Confidence intervals vs Bayesian intervals. In W. L. Harper & C. A. Hooker (Eds.), *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science, Vol. II* (pp. 175–257). Dordrecht, Holland: D. Reidel Publishing Company.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge, UK: Cambridge University Press.

Jefferys, W. H., & Berger, J. O. (1992). Ockham's razor and Bayesian analysis. *American Scientist*, *80*, 64–72.

Jeffreys, H. (1961). *Theory of Probability* (3rd edn.). Oxford, UK: Oxford University Press.

Jones, M., & Love, B. (2011). Bayesian fundamentalism or enlightenment? On the explanatory status and theoretical contributions of Bayesian models of cognition. *Behavioral and Brain Sciences*, *34*, 169–231.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, *90*, 377–395.

Kass, R. E., & Wasserman, L. (1995). A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. *Journal of the American Statistical Association*, *90*, 928–934.

Kass, R. E., & Wasserman, L. (1996). The selection of prior distributions by formal rules. *Journal of the American Statistical Association*, *91*, 1343–1370.

Katsikopoulos, K. V., Schooler, L. J., & Hertwig, R. (2010). The robust beauty of ordinary information. *Psychological Review*, *117*, 1259-1266.

Klauer, K. (2010). Hierarchical multinomial processing tree models: A latent-trait approach. *Psychometrika*, *75*, 70–98.

Kraemer, H. C. (1992). Measurement of reliability for categorical data in medical research. *Statistical Methods in Medical Research*, *1*, 183–199.

Kraemer, H. C., Periyakoil, V. S., & Noda, A. (2004). Kappa coefficients in medical research. In R. B. D'Agostino (Ed.), *Tutorials in Biostatistics, Vol. 1: Statistical Methods in Clinical Studies.* New York: John Wiley.

Kruschke, J. K. (1993). Human category learning: Implications for backpropagation models. *Connection Science*, *5*, 3–36.

Kruschke, J. K. (2010a). *Doing Bayesian Data Analysis: A Tutorial Introduction with R and BUGS*. Burlington, MA: Academic Press.

Kruschke, J. K. (2010b). What to believe: Bayesian methods for data analysis. *Trends in Cognitive Sciences*, *14*, 293–300.

Kuss, M., Jäkel, F., & Wichmann, F. A. (2005). Bayesian inference for psychometric functions. *Journal of Vision*, *5*, 478–492.

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*, 159–174.

Lasserre, J., Bishop, C. M., & Minka, T. (2006). Principled hybrids of generative and discriminative models. In *Proceedings 2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 87–94). Washington, DC: IEEE Computer Society.

Le Corre, M., & Carey, S. (2007). One, two, three, four, nothing more: An investigation of the conceptual sources of the verbal counting principles. *Cognition*, *105*, 395–438.

Lee, M. D. (2011a). How cognitive modeling can benefit from hierarchical Bayesian models. *Journal of Mathematical Psychology*, *55*, 1–7.

Lee, M. D. (2011b). In praise of ecumenical Bayes. *Behavioral and Brain Sciences*, *34*, 206–207.

Lee, M. D., & Cummins, T. D. R. (2004). Evidence accumulation in decision making: Unifying the "take the best" and "rational" models. *Psychonomic Bulletin & Review*, *11*, 343–352.

Lee, M. D., & Pooley, J. P. (2013). Correcting the SIMPLE model of free recall. *Psychological Review*, *120*, 293–296.

Lee, M. D., & Sarnecka, B. W. (2010). A model of knower-level behavior in number concept development. *Cognitive Science*, *34*, 51–67.

Lee, M. D., & Sarnecka, B. W. (2011). Number knower-levels in young children: Insights from a Bayesian model. *Cognition*, *120*, 391–402.

Lehrner, J. P., Kryspin-Exner, I., & Vetter, N. (1995). Higher olfactory threshold and decreased odor identification ability in HIV-infected persons. *Chemical Senses*, *20*, 325–328.

Leigh, B. C., & Stall, R. (1993). Substance use and risky sexual behavior for exposure to HIV: Issues in methodology, interpretation, and prevention. *American*

Psychologist, *48*, 1035–1045.

Lejuez, C. W., Read, J. P., Kahler, C. W., Richards, J. B., Ramsey, S. E., Stuart, G. L., et al. (2002). Evaluation of a behavioral measure of risk taking: The balloon analogue risk task (BART). *Journal of Experimental Psychology: Applied*, *8*, 75–84.

Leung, A. K.-Y., Kim, S., Polman, E., Ong, L. S., Qiu, L., Goncalo, J. A., et al. (2012). Embodied metaphors and creative "acts". *Psychological Science*, *23*, 502–509.

Lewis, S. M., & Raftery, A. E. (1997). Estimating Bayes factors via posterior simulation with the Laplace–Metropolis estimator. *Journal of the American Statistical Association*, *92*, 648–655.

Liang, F., Paulo, R., Molina, G., Clyde, M. A., & Berger, J. O. (2008). Mixtures of g priors for Bayesian variable selection. *Journal of the American Statistical Association*, *103*, 410–423.

Lindley, D. V. (1972). *Bayesian Statistics, A Review*. Philadelphia, PA: SIAM.

Lindley, D. V. (1986). Comment on "Why Isn't Everyone a Bayesian?" by Bradley Efron. *The American Statistician*, *40*, 6–7.

Lindley, D. V. (1993). The analysis of experimental data: The appreciation of tea and wine. *Teaching Statistics*, *15*, 22–25.

Lindley, D. V. (2000). The philosophy of statistics. *The Statistician*, *49*, 293–337.

Liu, C. C., & Aitkin, M. (2008). Bayes factors: Prior sensitivity and model generalizability. *Journal of Mathematical Psychology*, *52*, 362–375.

Liu, J., & Wu, Y. (1999). Parameter expansion for data augmentation. *Journal of the American Statistical Association*, *94*, 1264–1274.

Lodewyckx, T., Kim, W., Tuerlinckx, F., Kuppens, P., Lee, M. D., & Wagenmakers, E.-J. (2011). A tutorial on Bayes factor estimation with the product space method. *Journal of Mathematical Psychology*, *55*, 331–347.

Lunn, D. J. (2003). WinBUGS development interface (WBDev). *ISBA Bulletin*, *10*, 10–11.

Lunn, D. J., Spiegelhalter, D., Thomas, A., & Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, *28*, 3049–3067.

Lunn, D. J., Thomas, A., Best, N., & Spiegelhalter, D. (2000). WinBUGS – a Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, *10*, 325–337.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge, MA: Cambridge University Press.

MacMillan, N., & Creelman, C. D. (2004). *Detection Theory: A User's Guide* (2nd edn.). Hillsdale, NJ: Erlbaum.

Marr, D. C. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco, CA: W. H. Freeman.

Masson, M. E. J. (2011). A tutorial on a practical Bayesian alternative to null-hypothesis significance testing. *Behavior Reseach Methods*, *43*, 679–690.

McEwan, R. T., McCallum, A., Bhopal, R. S., & Madhok, R. (1992). Sex and the risk of HIV infection: The role of alcohol. *British Journal of Addiction*, *87*, 577–584.

Merkle, E. C., Smithson, M., & Verkuilen, J. (2011). Hierarchical models of simple mechanisms underlying confidence in decision making. *Journal of Mathematical Psychology*, *55*, 57–67.

Murdock, B. B., Jr. (1962). The serial position effect in free recall. *Journal of Experimental Psychology*, *64*, 482–488.

Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, *47*, 90–100.

Myung, I. J., & Pitt, M. A. (1997). Applying Occam's razor in modeling cognition: A Bayesian approach. *Psychonomic Bulletin & Review*, *4*, 79–95.

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *10*, 104–114.

Nosofsky, R. M. (1986). Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Psychology: General*, *115*, 39–57.

Ntzoufras, I. (2009). *Bayesian Modeling Using WinBUGS*. Hoboken, NJ: John Wiley.

O'Hagan, A. (1995). Fractional Bayes factors for model comparison. *Journal of the Royal Statistical Society, Series B*, *57*, 99–138.

O'Hagan, A., & Forster, J. (2004). *Kendall's Advanced Theory of Statistics, Vol. 2B: Bayesian Inference* (2nd edn.). London: Arnold.

Ortega, A., Wagenmakers, E.-J., Lee, M. D., Markowitsch, H. J., & Piefke, M. (2012). A Bayesian latent group analysis for detecting poor effort in the assessment of malingering. *Archives of Clinical Neuropsychology*, *27*, 453–465.

Parsons, L. M., & Osherson, D. (2001). New evidence for distinct right and left brain systems for deductive and probabilistic reasoning. *Cerebral Cortex*, *11*, 954–965.

Payne, J. W., Bettman, J. R., & Johnson, E. J. (1990). *The Adaptive Decision Maker*. New York: Cambridge University Press.

Piaget, J. (1952). *The Child's Conception of Number*. New York: Norton.

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*. Vienna, Austria.

Poehling, K. A., Griffin, M. R., & Dittus, R. S. (2002). Bedside diagnosis of influenza virus infections in hospitalized children. *Pediatrics*, *110*, 83–88.

Poirier, D. J. (2006). The growth of Bayesian methods in statistics and economics since 1970. *Bayesian Analysis*, *1*, 969–980.

Ratcliff, R., & McKoon, G. (1997). A counter model for implicit priming in perceptual word identification. *Psychological Review*, *104*, 319–343.

Riefer, D. M., Knapp, B. R., Batchelder, W. H., Bamber, D., & Manifold, V. (2002). Cognitive psychometrics: Assessing storage and retrieval deficits in special populations with multinomial processing tree models. *Psychological Assessment*, *14*, 184–201.

Rieskamp, J. (2008). The probabilistic nature of preferential choice. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *34*, 1446–1465.

Rips, L. J. (2001). Two kinds of reasoning. *Psychological Science*, *12*, 129–134.

Rolison, J. J., Hanoch, Y., & Wood, S. (2012). Risky decision making in younger and older adults: The role of learning. *Psychology and Aging*, *27*, 129–140.

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review*, *16*, 225–237.

Rubin, D. C., Hinton, S., & Wenzel, A. (1999). The precise time course of retention. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *25*, 1161–1176.

Rubin, D. C., & Wenzel, A. E. (1996). One hundred years of forgetting: A quantitative description of retention. *Psychological Review*, *103*, 734–760.

Sanborn, A. N., Griffiths, T. L., & Shiffrin, R. M. (2010). Uncovering mental representations with Markov chain Monte Carlo. *Cognitive Psychology*, *60*, 63–106.

Schaeffer, B., Eggleston, V. H., & Scott, J. L. (1974). Number development in young children. *Cognitive Psychology*, *6*, 357–379.

Scheibehenne, B., Rieskamp, J., & Wagenmakers, E.-J. (2013). Testing adaptive toolbox models: A Bayesian hierarchical approach. *Psychological Review*, *120*, 39–64.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Sellke, T., Bayarri, M. J., & Berger, J. O. (2001). Calibration of p values for testing precise null hypotheses. *The American Statistician*, *55*, 62–71.

Shepard, R. N., Hovland, C. I., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs*, *75*, 1–42.

Sheu, C.-F., & O'Curry, S. L. (1998). Simulation-based Bayesian inference using BUGS. *Behavior Research Methods, Instruments, & Computers*, *30*, 232–237.

Shrout, P. E. (1998). Measurement reliability and agreement in psychiatry. *Statistical Methods in Medical Research*, *7*, 301–317.

Sisson, S. A. (2005). Transdimensional Markov chains: A decade of progress and future perspectives. *Journal of the American Statistical Association*, *100*, 1077–1089.

Sloman, S. A. (1998). Categorical inference is not a tree: The myth of inheritance hierarchies. *Cognitive Psychology*, *35*, 1–33.

Smith, A. F. M., & Spiegelhalter, D. J. (1980). Bayes factors and choice criteria for linear models. *Journal of the Royal Statistical Society, Series B*, *42*, 213–220.

Smith, J. B., & Batchelder, W. H. (2010). Beta-MPT: Multinomial processing tree models for addressing individual differences. *Journal of Mathematical*

Psychology, *54*, 167–183.

Smithson, M. (2010). A review of six introductory texts on Bayesian methods. *Journal of Educational and Behavioral Statistics*, *35*, 371–374.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & Van Der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B*, *64*, 583–639.

Stan Development Team. (2013). *Stan: A C++ Library for Probability and Sampling, Version 1.3*.

Stone, C. J., Hansen, M. H., Kooperberg, C., & Truong, Y. K. (1997). Polynomial splines and their tensor products in extended linear modeling (with discussion). *The Annals of Statistics*, *25*, 1371–1470.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., & Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, *331*, 1279–1285.

Townsend, J. T. (1975). The mind–body equation revisited. In C. Cheng (Ed.), *Philosophical Aspects of the Mind–Body Problem* (pp. 200–218). Honolulu, HI: Honolulu University Press.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460.

Uebersax, J. S. (1987). Diversity of decision-making models and the measurement of interrater agreement. *Psychological Bulletin*, *101*, 140–146.

van Ravenzwaaij, D., Dutilh, G., & Wagenmakers, E.-J. (2011). Cognitive model decomposition of the BART: Assessment and application. *Journal of Mathematical Psychology*, *55*, 94–105.

Vandekerckhove, J., Tuerlinckx, F., & Lee, M. D. (2011). Hierarchical diffusion models for two-choice response time. *Psychological Methods*, *16*, 44–62.

Vanpaemel, W. (2010). Prior sensitivity in theory testing: An apologia for the Bayes factor. *Journal of Mathematical Psychology*, *54*, 491–498.

Verdinelli, I., & Wasserman, L. (1995). Computing Bayes factors using a generalization of the Savage–Dickey density ratio. *Journal of the American Statistical Association*, *90*, 614–618.

Wagenmakers, E.-J. (2007). A practical solution to the pervasive problems of p values. *Psychonomic Bulletin & Review*, *14*, 779–804.

Wagenmakers, E.-J., Lee, M. D., Lodewyckx, T., & Iverson, G. (2008). Bayesian versus frequentist inference. In H. Hoijtink, I. Klugkist, & P. A. Boelen (Eds.), *Bayesian Evaluation of Informative Hypotheses* (pp. 181–207). New York: Springer Verlag.

Wagenmakers, E.-J., & Morey, R. D. (2013). Simple relation between one–sided and two–sided Bayesian point–null hypothesis tests. *Manuscript submitted for publication*.

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., & van der Maas, H. L. J. (2011). Why psychologists must change the way they analyze their data: The case of psi. *Journal of Personality and Social Psychology*, *100*, 426–432.

Wagenmakers, E.-J., Wetzels, R., Borsboom, D., van der Maas, H. L. J., & Kievit, R. A. (2012). An agenda for purely confirmatory research. *Perspectives on Psychological Science*, *7*, 627–633.

Wallsten, T. S., Pleskac, T. J., & Lejuez, C. W. (2005). Modeling behavior in a clinically diagnostic sequential risk-taking task. *Psychological Review*, *112*, 862–880.

Wetzels, R., Grasman, R. P. P. P., & Wagenmakers, E.-J. (2010). An encompassing prior generalization of the Savage–Dickey density ratio test. *Computational Statistics & Data Analysis*, *54*, 2094–2102.

Wetzels, R., Lee, M. D., & Wagenmakers, E.-J. (2010). Bayesian inference using WBDev: A tutorial for social scientists. *Behavior Research Methods*, *42*, 884–897.

Wetzels, R., Vandekerckhove, J., Tuerlinckx, F., & Wagenmakers, E. (2010). Bayesian parameter estimation in the Expectancy Valence model of the Iowa gambling task. *Journal of Mathematical Psychology*, *54*, 14–27.

Wynn, K. (1990). Children's understanding of counting. *Cognition*, *36*, 155–193.

Wynn, K. (1992). Children's acquisition of number words and the counting system. *Cognitive Psychology*, *24*, 220–251.

Zeelenberg, R., Wagenmakers, E.-J., & Raaijmakers, J. G. W. (2002). Priming in implicit memory tasks: Prior study causes enhanced discriminability, not only bias. *Journal of Experimental Psychology: General*, *131*, 38–47.

Zeigenfuse, M. D., & Lee, M. D. (2010). A general latent-assignment approach for modeling psychological contaminants. *Journal of Mathematical Psychology*, *54*, 352–362.

Zellner, A., & Siow, A. (1980). Posterior odds ratios for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian Statistics* (pp. 585–603). Valencia: University Press.

Index