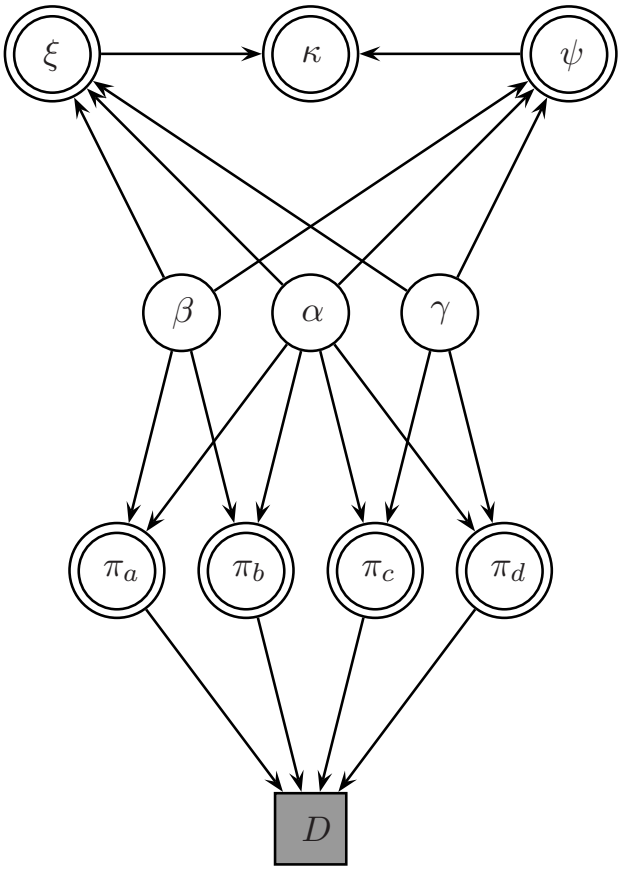
A Course in Bayesian Graphical Modeling for Cognitive Science

Michael D. Lee
University of California, Irvine
mdlee@uci.edu

Eric-Jan Wagenmakers
University of Amsterdam
ej.wagenmakers@gmail.com

August 23, 2010



This publication is typeset in LaTeXusing the Memoir class.

CONTENTS

PREFACE

This book 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 Bayesian models, apply them to your own data, and draw your own conclusions.

This book is based on three principles. The first is that of *accessability*: 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, one that we feel keeps students interested and motivated to learn more.

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 trivial to first construct a basic model, and then add individual differences as random effects or separate groups, add substantive prior information, add covariates, add a contaminant process, etc. In other words, Bayesian modeling is flexible and respects the complexities that are inherent in the modeling of cognitive phenomena.

This book has been used in master courses at the University of California at Irvine and at the University of Amsterdam. We have found that students master the material quickly, and that they appreciate a statistical framework in which inference is coherent and rational, even if they might not acknowledge this explicitly.

We hope that after completing this course, 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.

We like to thank John Miyamoto for constructive criticism and suggestions for improvement, and Dora Matzke for her help in programming and plotting.

Michael D. Lee
Irvine, USA
Eric-Jan Wagenmakers
Amsterdam, The Netherlands
August 2010

Part I

Parameter Estimation for Statistical Models

CHAPTER 1

BAYESIAN PARAMETER ESTIMATION

As is customary, we introduce Bayesian parameter estimation by means of the binomial example. Assume we prepare for you a series of 10 factual true/false questions of equal difficulty. Interest centers on your latent probability θ of answering any one question correctly. In Bayesian inference, uncertainty with respect to parameters is—at any point in time—quantified by probability distributions. Thus, in order to get the Bayesian inference machine off the ground, we need to specify our uncertainty with respect to θ before seeing the data. Suppose you do not know anything about 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 proceed with the test, and find that you answered 9 out of 10 questions correctly. After having seen these data, our updated knowledge about θ is described by a "posterior distribution", denoted $p(\theta|s, n)$, where $s = 9$ and $n = 10$ indicate the number of successes and the number of questions, respectively. Assume that the probability of the data is given by the binomial distribution:

$$p(s|\theta, n) = \binom{n}{s}\theta^s(1-\theta)^{n-s}. \tag{1.1}$$

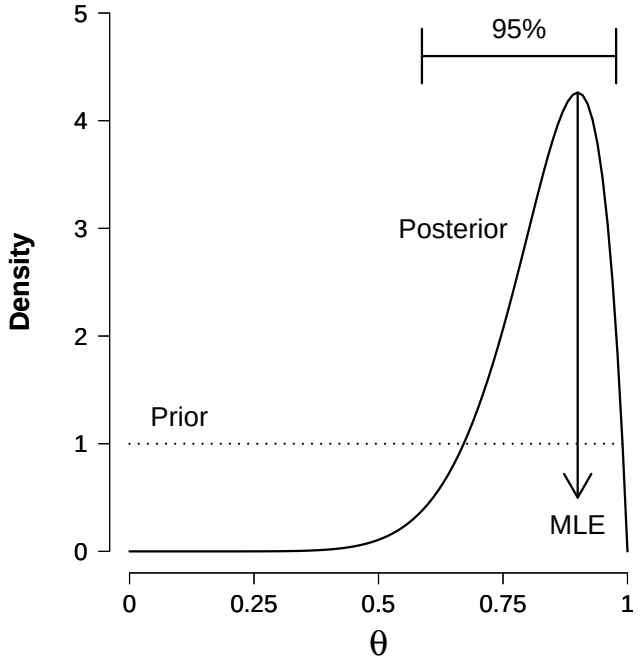


Figure 1.1: Bayesian parameter estimation for binomial rate parameter θ, after observing 9 correct responses and 1 incorrect response. The mode of the posterior distribution for θ is 0.9, equal to the maximum likelihood estimate, and the 95% confidence interval extends from 0.59 to 0.98.

The transition from prior $p(\theta)$ to posterior $p(\theta|s,n)$ is then given by Bayes' rule,

$$p(\theta|s,n) = \frac{p(s|\theta,n)p(\theta)}{p(s|n)}. \tag{1.2}$$

This equation is often verbalized as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}. \tag{1.3}$$

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 the area under the posterior distribution equals 1. Therefore, Equation 1.2 is often written as

$$p(\theta|s,n) \propto p(s|\theta,n)p(\theta), \tag{1.4}$$

which says that the posterior is proportional to the likelihood times the prior.

The solid line in Figure 1.1 shows the posterior distribution for θ, which is obtained when the uniform prior is updated with data $s = 9$ and $n = 10$. 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, $\hat{\theta} = s/n = 0.9$ (Myung, 2003). The spread of a posterior distribution is most easily captured by a Bayesian x% confidence interval that extends from the $(x/2)$th to the $(100 - x/2)$th percentile of the posterior distribution. For the posterior distribution in Figure 1.1, a 95% Bayesian confidence interval for θ extends from 0.59 to 0.98. In contrast to the classical or 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.

Now suppose we design a new set of 5 questions, of equal 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 $p(\theta|n = 10, s = 9)$— 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 $n^{rep} = 5$ questions? The mathematical solution is to integrate over the posterior, $\int p(s^{rep}|\theta, n^{rep} = 5)p(\theta|n = 10, s = 9)d\theta$, where s^{rep} is the predicted number of correct responses out of the additional set of 5 questions. Computationally, one may 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 s_i^{rep} by means of Equation 1.1. The end result is $p(s^{rep})$, the predictive density 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. In contrast, much of classical inference relies on the "plug–in principle" that in this case would lead us to predict $p(s^{rep})$ solely based on $\hat{\theta}$, the maximum likelihood estimate. Plug–in procedures ignore uncertainty in θ, and hence lead to predictions that are overconfident, that is, predictions that are less variable than they should be (Aitchison & Dunsmore, 1975).[1]

You are now presented with the new set of 5 questions. You answer 3 out of 5 correctly. How do 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 precise order in which

[1] It should be acknowledged that classical statisticians can account for uncertainty in the estimation of θ by repeatedly drawing a bootstrap sample from the data, calculating the associated bootstrap MLE, and finding the corresponding prediction for s^{rep} (e.g., Wagenmakers, Ratcliff, Gomez, & Iverson, 2004).

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. This again contrasts with classical inference, in which inference for sequential designs is radically different from that for non–sequential designs (for a discussion, see e.g., 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. Unfortunately, the posterior distribution or any of its summary measures can only be obtained in closed form for a restricted set of relatively simple models. To illustrate in the case of our binomial example, the uniform prior is a so–called 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, be it with parameters $\alpha + s$ and $\beta + n - s$. In simple *conjugate* cases such as these, where the prior and the posterior belong to the same distributional family, it is possible to obtain closed form solutions for the posterior distribution, but in other more interesting cases it is not.

For a long time, researchers did not know how to proceed with Bayesian inference when the posterior could not be obtained in close form. 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 (i.e., MCMC; e.g., Gamerman & Lopes, 2006; Gilks, Richardson, & Spiegelhalter, 1996). Using MCMC techniques such as Gibbs sampling or the Metropolis–Hastings algorithm, researchers can now directly sample sequences of values from the posterior distribution of interest, foregoing the need for closed form analytic solutions. At the time of writing, the adage is that Bayesian models are limited only by the user's imagination.

To provide a concrete and simple illustration of Bayesian inference using MCMC, we revisit our binomial example of 9 correct responses out of 10 questions, and the associated inference problem for θ, the probability of answering any one question correctly. Throughout this article, we use the general–purpose WinBUGS program (Lunn, Thomas, Best, & Spiegelhalter, 2000; an introduction for psychologists is given by Sheu & O'Curry, 1998) that allows the user to specify and fit models without having to hand–code the MCMC algorithms. Although WinBUGS does not work for every application, it will work for most applications in the field of psychology. The WinBUGS program is easy to learn and is supported by a large community of active researchers.

The WinBUGS program requires the user 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 prior and the posterior of θ. The associated WinBUGS model specification code is three lines long:

```
model
{
  theta ~ dbeta(1,1) # the uniform prior for updating by the data
  k ~ dbin(theta,n)  # the data; in our example, k = 9 and n = 10
  thetaprior ~ dbeta(1,1) # a uniform prior not for updating
}
```

In this code, the "$\sim$" or twiddle symbol denotes "is distributed as", `dbeta(a,b)` indicates the beta distribution with parameters a and b, and `dbin(theta,n)` indicates the binomial distribution with rate theta and n observations. These and many other distributions are build in to the WinBUGS

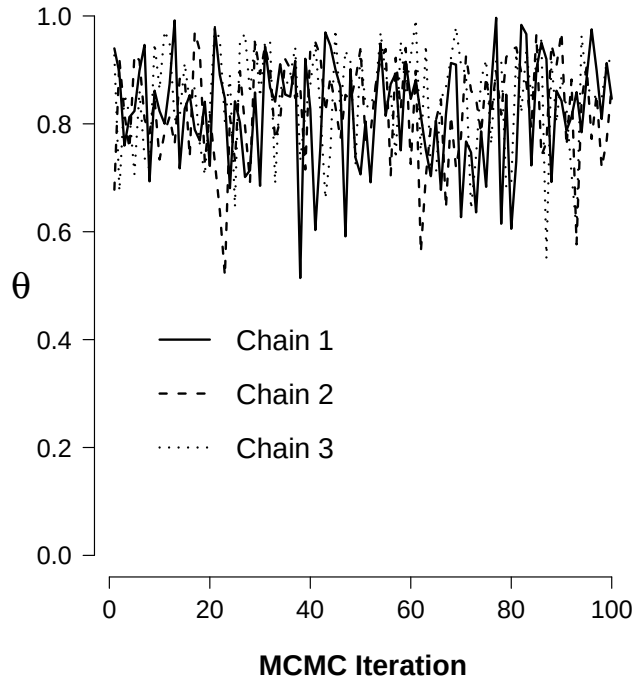


Figure 1.2: Three MCMC chains for binomial rate parameter θ, after observing 9 correct responses and 1 incorrect response.

system. The "#" or hash sign is used for commenting out what should not be compiled. As WinBUGS is a declarative language, the order of the three lines is inconsequential.

When this code is executed, the user obtains a sequence of samples (i.e., an MCMC chain) from the posterior $p(\theta|D)$ and a sequence of samples from the prior $p(\theta)$. In more complex models, it may take some time before the 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 are hence unaffected by the starting values) it is good practice to discard the first samples as "burn–in", and to diagnose convergence by running multiple chains.

For instance, Figure 1.2 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. Quantitative measures for diagnosing convergence are also available (e.g., the Gelman and Rubin (1992) $\hat{R}$ statistic, that compares within–chain to between–chain variability; for more recommendations regarding convergence see e.g., Gelman, 1996; Gelman & Hill, 2007).

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 1.2 were run for 3000 iterations each, for a total of 9000 samples for the prior and the posterior of θ. Figure 1.3 plots histograms[2] for the prior (i.e., dotted line) and the posterior (i.e., thick solid line). To visualize how the histograms are constructed from the MCMC chains, the bottom panel of Figure 1.3 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 1.3, the thin solid lines represent logspline nonparametric density estimates (Stone, Hansen, Kooperberg, & Truong, 1997). The mode of the logspline density esti-

[2]These histograms were constructed such that the total area under each histogram equals one.

mate for the posterior of θ is 0.89, whereas the 95% confidence interval is $(0.59, 0.98)$, matching the analytical result shown in Figure 1.1.

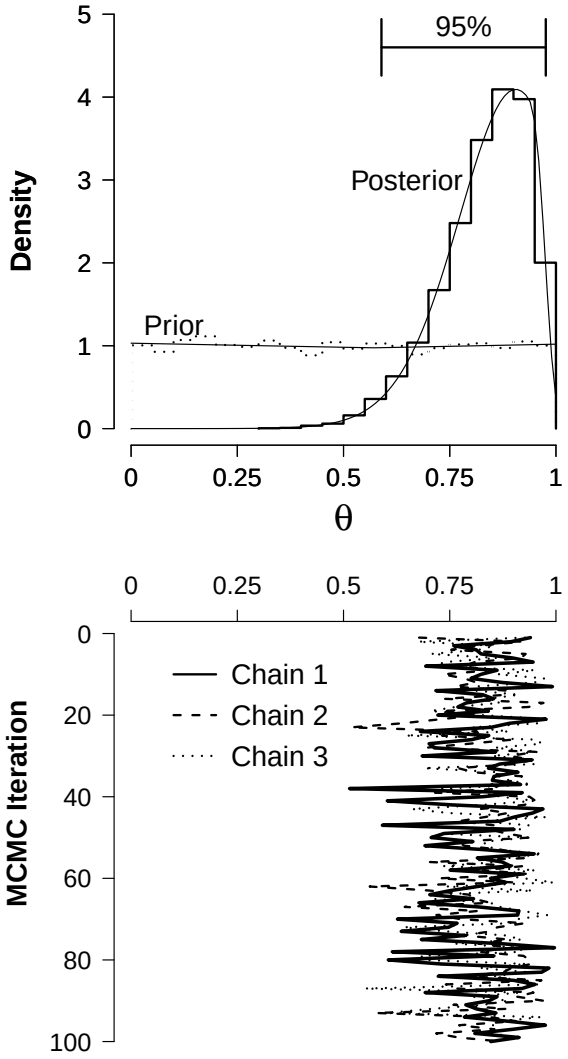


Figure 1.3: MCMC–based Bayesian parameter estimation for binomial rate parameter θ, after observing 9 correct responses and 1 incorrect response. The thin solid lines indicate the fit of a nonparametric logspline density estimator. Based on this density estimator, the mode of the posterior distribution for θ is approximately 0.89, and the 95% confidence interval extends from 0.59 to 0.98, closely matching the analytical results from Figure 1.1.

Of course, this example represents an ideal scenario; in more complicated models, convergence might be obtained only after many MCMC iterations – that is, chains may move very slowly from their starting point to the stationary distribution, and even after they get there the chains may take a long time to collect enough independent samples. This problem is often easy to diagnose

by running multiple chains and by computing the correlations between successive samples – a high autocorrelation suggests slow convergence or slow mixing. Thus, for complex models it is important to use MCMC algorithms that are efficient, reliable, and quick. This is currently an active area of research. Nevertheless, the fundamental theoretical obstacles for Bayesian parameter estimation have been overcome. In fields such as statistics, artificial intelligence, and machine learning, MCMC algorithms are now used on a routine basis.

CHAPTER 2

GETTING STARTED WITH WINBUGS

WITH DORA MATZKE

Throughout this course book, you will use the WinBUGS (Lunn et al., 2000) 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.

2.1 Installing WinBUGS, Matbugs, and R2WinBugs

Installing WinBUGS

WinBUGS is a 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 go to the effort of downloading and applying the registration key. Some of the exercises in this course 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`.

Installing Matlab and Matbugs

Matlab is a commercial software, and is available at `http://www.mathworks.com/`. As best we know, any reasonably recent version of Matlab should let you do the exercises in this course. Also, as best 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 `http://www.cs.ubc.ca/~murphyk/Software/MATBUGS/matbugs.html`.

Installing R and R2WinBUGS

R is a free software, and is available at `http://www.r-project.org/`. You can download the Windows version of R directly from `http://cran.nedmirror.nl/bin/windows/base/`. 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 chose your preferred CRAN mirror (e.g., Netherlands (Amsterdam 2)), select `R2WinBUGS` in the `Packages` window and click on `OK`.

2.2 Using the Applications

An Example with the Binomial Distribution

We will illustrate the use of WinBUGS, Matbugs, and R by means of a simple example involving a binary process. A binary process is anything where there are only two possible outcomes. It might be that something either happens or does not happen, or that something either succeeds or fails, or that something takes one value rather than the other. 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 values (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 values are produced. Assuming that the trials are statistically independent (i.e., what happened on one trial does not influence the other trials), the number of successes k follows a Binomial distribution, $k \sim \text{Binomial}\big(\theta, n\big)$. This relationship means that by observing the k successes out of n trials, it is possible to update our knowledge 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.

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 counts of the numbers of successes k and the 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 on θ, $\theta \sim \text{Uniform}\big(0, 1\big)$.

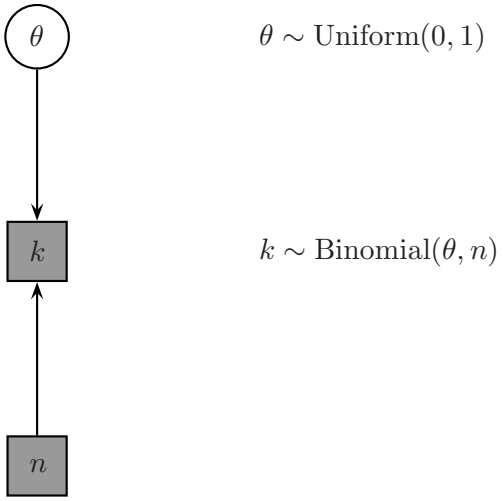


Figure 2.1: Graphical model for inferring the rate of a binary process.

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

Using WinBUGS

WinBUGS requires the user to construct a file that contains the data, a file that contains the starting values for the model parameters, and a file that contains the model specification. The WinBUGS model specification code associated with our binomial example is as follows:

```
model {
   # Prior on Rate
   theta ~ dbeta(1,1)
   # Observed Counts
   k ~ dbin(theta,n)
}
```

Note that, even though conceptually the prior on θ is Uniform$\big(0,1\big)$, it has been implemented as Beta$\big(1,1\big)$. These two distributions are the same, but WinBUGS seems to have fewer computational problems with the Beta distribution version.

To implement the model shown in Figure 2.1 and to obtain samples from the posterior distribution of θ, you need to work your way through the following steps.

1. Copy the model specification text above and paste it in a text file. Save the file, for instance 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. Don't 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 (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 (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 instance 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 (Figure 2.2). If the data is successfully loaded, the message "data is loaded" will appear in the status bar.

5. Set the number of chains. Multiple chains amount to multiple independent runs of the same model with the same data (although you can vary the starting point per chain), and so provide a key test of convergence—something we will discuss in more detail in a later chapter. 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 (Figure 2.2).

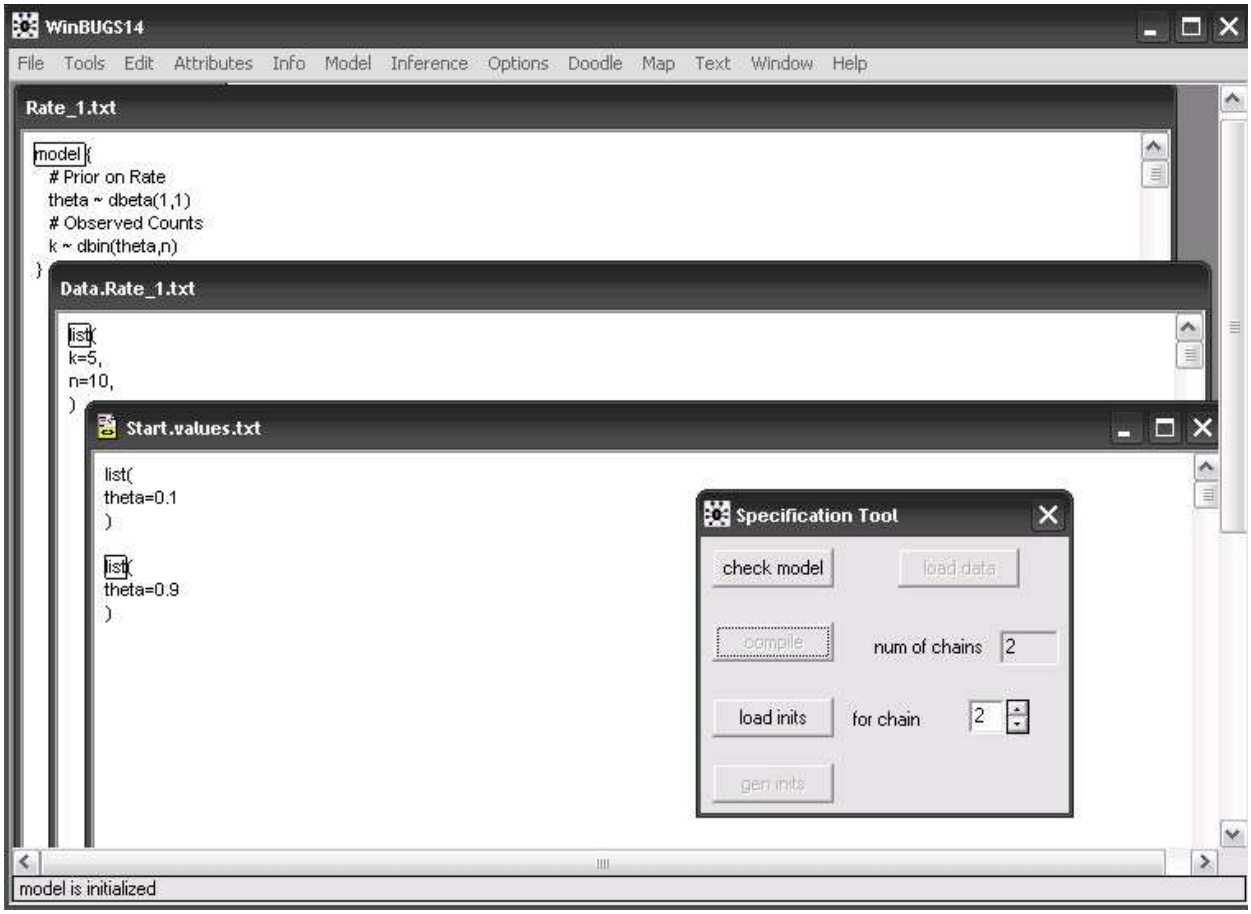


Figure 2.2: Model Specification Tool.

6. Compile the model. To compile the model, click on `compile` in the `Specification Tool` window (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., parameter θ). If you do not specify the starting values, WinBUGS will try to get them from the prior, which may or may not lead to numerical crashes. It is therefore safer to give a starting value to all unobserved variables (especially nodes that have no parents).

   The content of the file should look like this:

   ```
   list(
   theta=0.1
   )

   list(
   theta=0.9
   )
   ```

   Save the file, for instance 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 (Figure 2.2). To load the starting value for the second chain, highlight the second "list" command and click on `load inits` once again. If the 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 (Figure 2.3) is opened, type "theta" in the field labelled `node` and click on `set`.

10. Specify the number of samples you want to record. To this end, you first have to specify (1) the total number of samples you want to draw from the posterior of θ; and (2) 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 labelled `beg` (i.e., WinBUGS will start recording from the first sample) and type "20000" in the field labelled `end` in the `Sample Monitor Tool` window (Figure 2.3).

Figure 2.3: 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 (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. To this end, select the `Update` option from the `Model` menu. Once the `Update Tool` window (see 2.4) is opened, type "20000" in the field labelled `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 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 labelled `thin` in the `Update Tool` window (Figure 2.4).

14. Specify the number of samples after which WinBUGS should refresh its display. To this end, type "100" in the field labelled `refresh` in the `Update Tool` window (Figure 2.4).

15. Sample from the posterior. To sample from the posterior of θ, click on `update` in the `Update Tool` window (Figure 2.4). During sampling, the message "model updating" will appear in the status bar. Once the sampling is finished, the message "update took x secs" will appear in the status bar.

Figure 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 (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 (Figure 2.3). WinBUGS will paste the posterior distribution of θ in the log file.

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 that you took to specify and initialize the model. The first output item is the `Dynamic trace` plot that allows you to monitor θ 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. The second output item is the `Node statistics` table that presents the summary statistics for θ. Among others, 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.

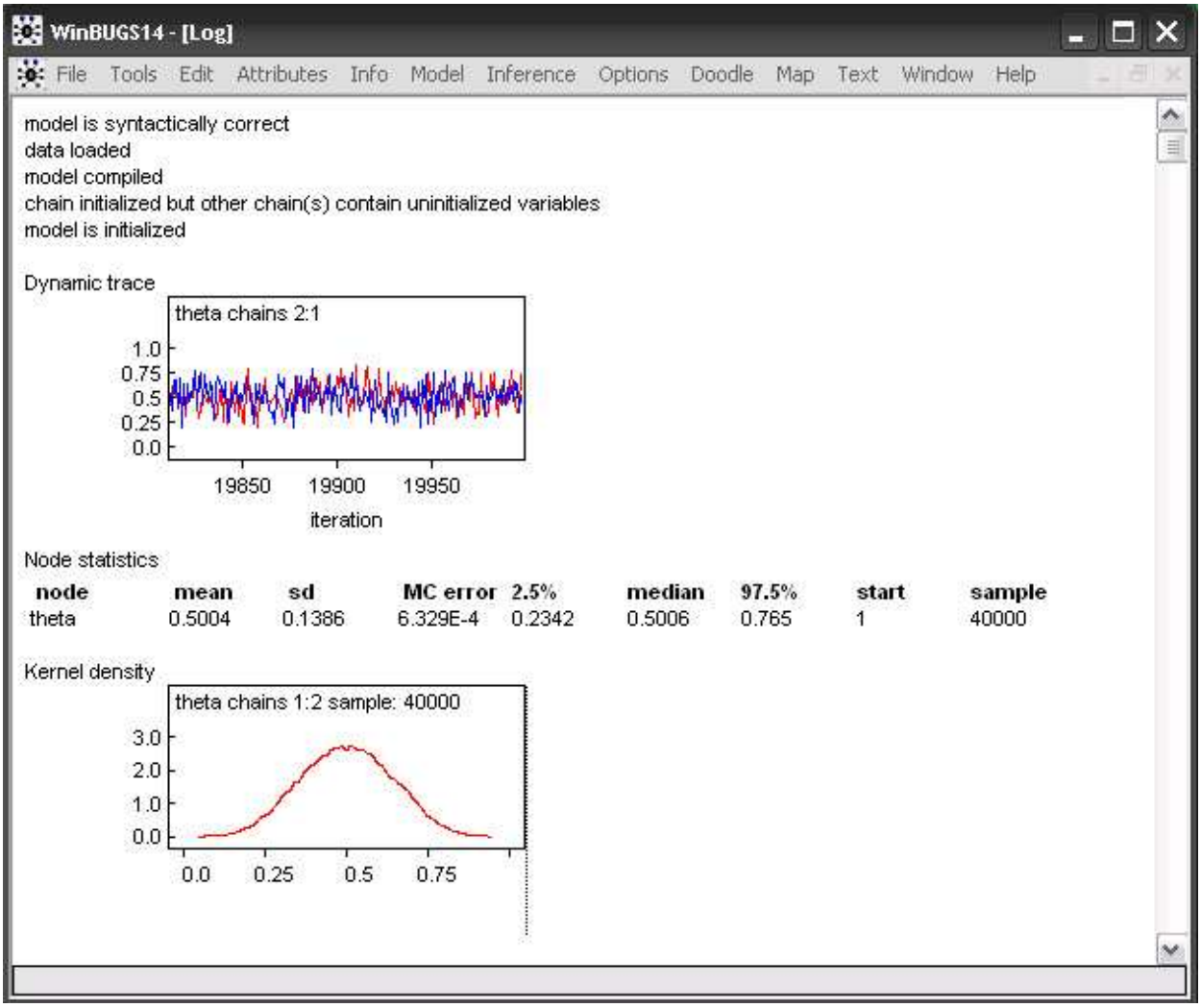


Figure 2.5: Example of an output log file.

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 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, ...

In one sense, it would be nice to understand exactly how WinBUGS managed to generate the posterior samples. In another sense, if you are interested in building and analyzing models and data, you do not necessarily need to understand the computational basis of posterior sampling (any more than you need to know how SPSS calculates a t-test statistic). If you understand the conceptual basis that underlies the generation of the posterior samples, you can happily build models and analyze data without worrying about the intricacies of Gibbs Sampling, Adaptive

Rejection Sampling, Markov-Chain Monte-Carlo, and all the rest.[1]

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 when it comes to debugging your WinBUGS code. The error messages are displayed in the bottom left corner of the status bar (the very small letters).

With respect to errors in the model specification, 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`):

```
model {
  #Prior on Rate
  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`" (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`" (Figure 2.7).

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`" (Figure 2.8).

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 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`" (Figure 2.9).

Exercises

Once you got WinBUGS and the scripts up and running, here is a list of exercises to complete. Try and think of the message each exercise has for making inferences about psychological variables from data.

[1] Some students find this 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 by Griffiths, Kemp, and Tenenbaum (2008). Continue with the relevant chapters in the excellent book by MacKay (2003), which is freely available on the Web, and follow the more technical references from there.

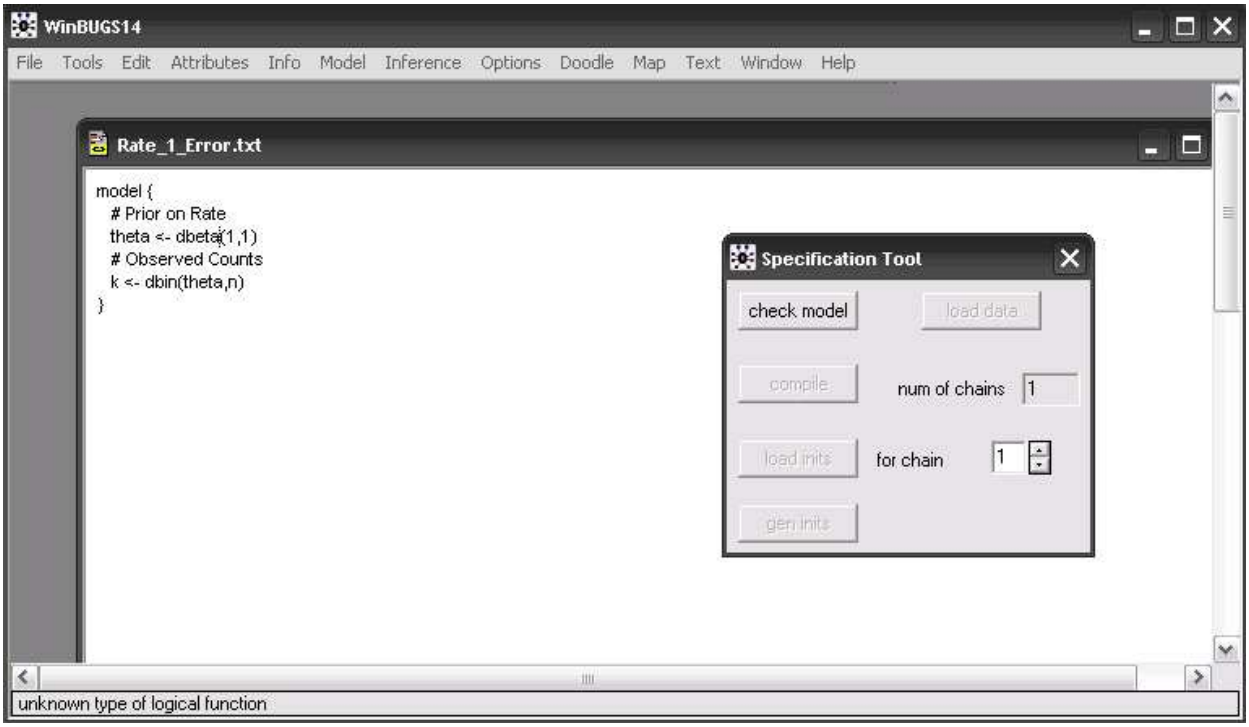


Figure 2.6: WinBUGS error message as a result of incorrect logical operators.

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

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

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

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 below (see also `Rate_1.m`).

```
% Set the working directory
cd D:\WinBUGS_Book\Matlab_codes

% 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?
```

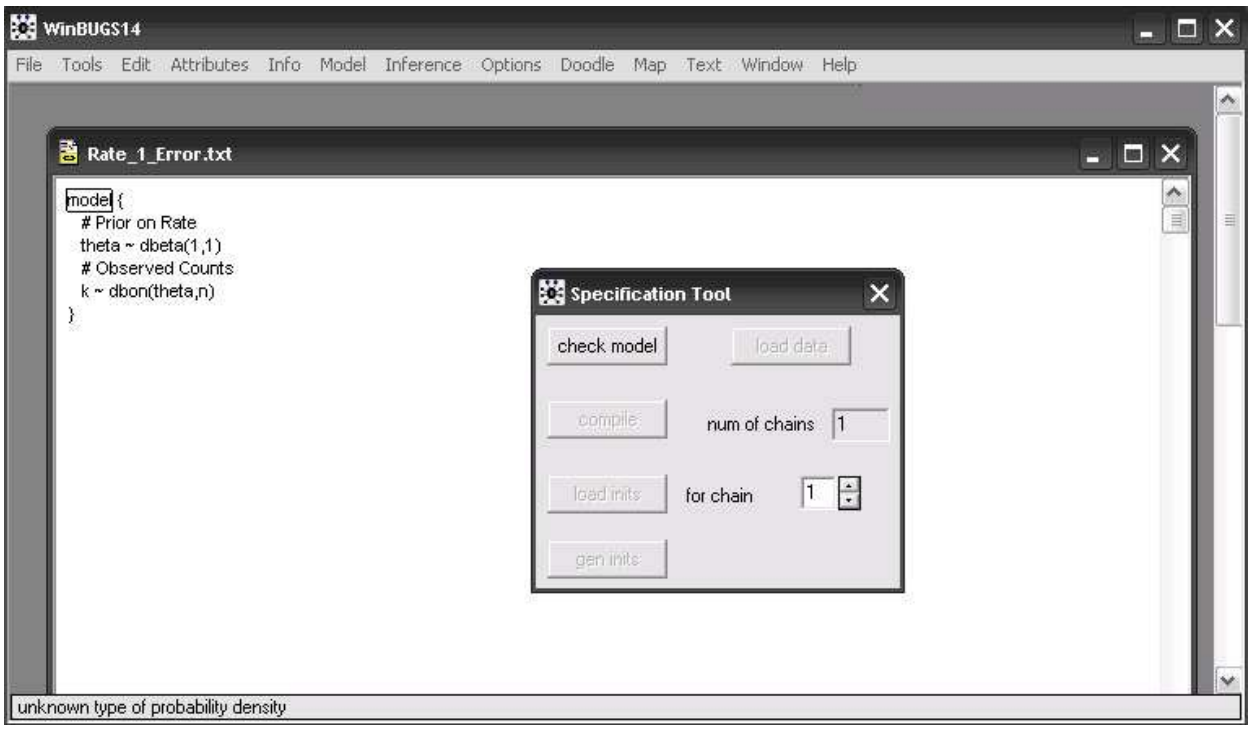


Figure 2.7: WinBUGS error message as a result of a misspecified probability density.

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

Some of these options 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).

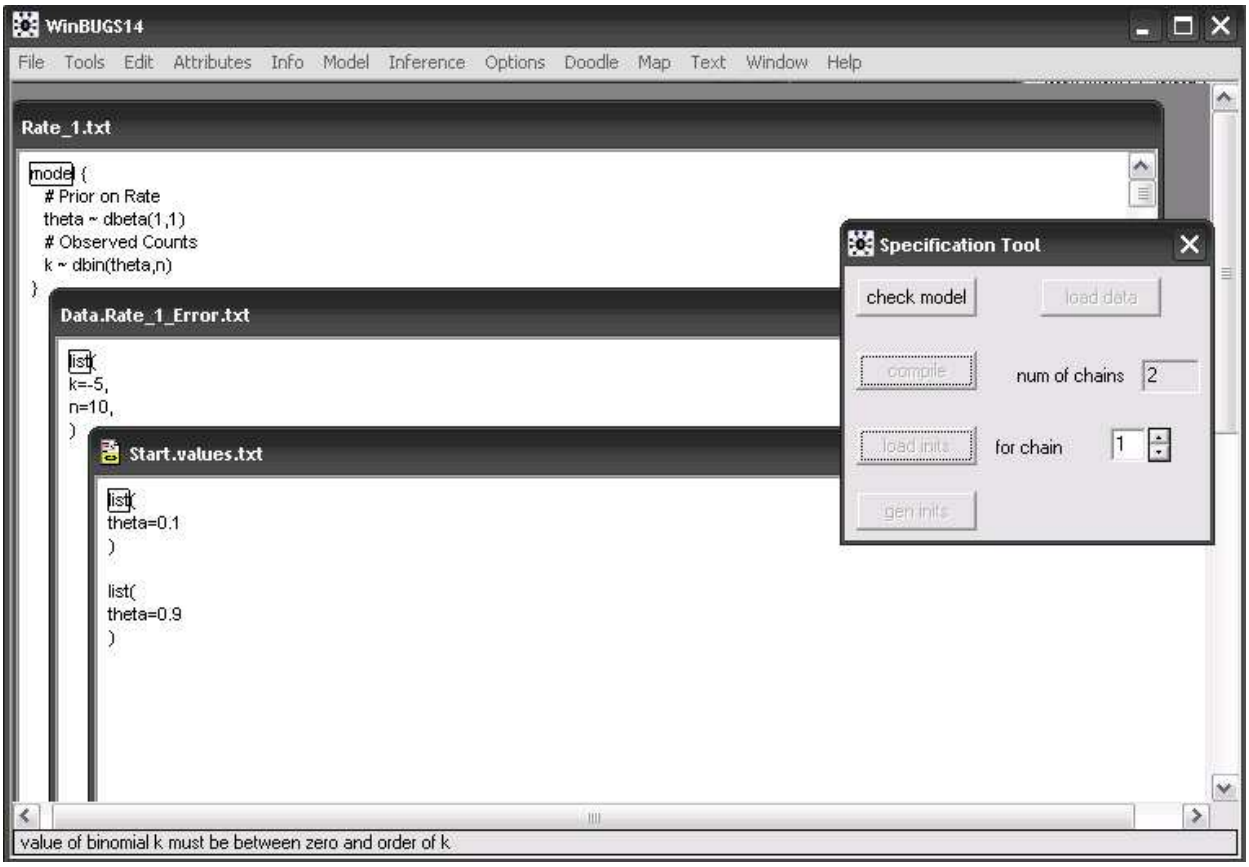


Figure 2.8: WinBUGS error message as a result of incorrect data.

- `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.

The 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.

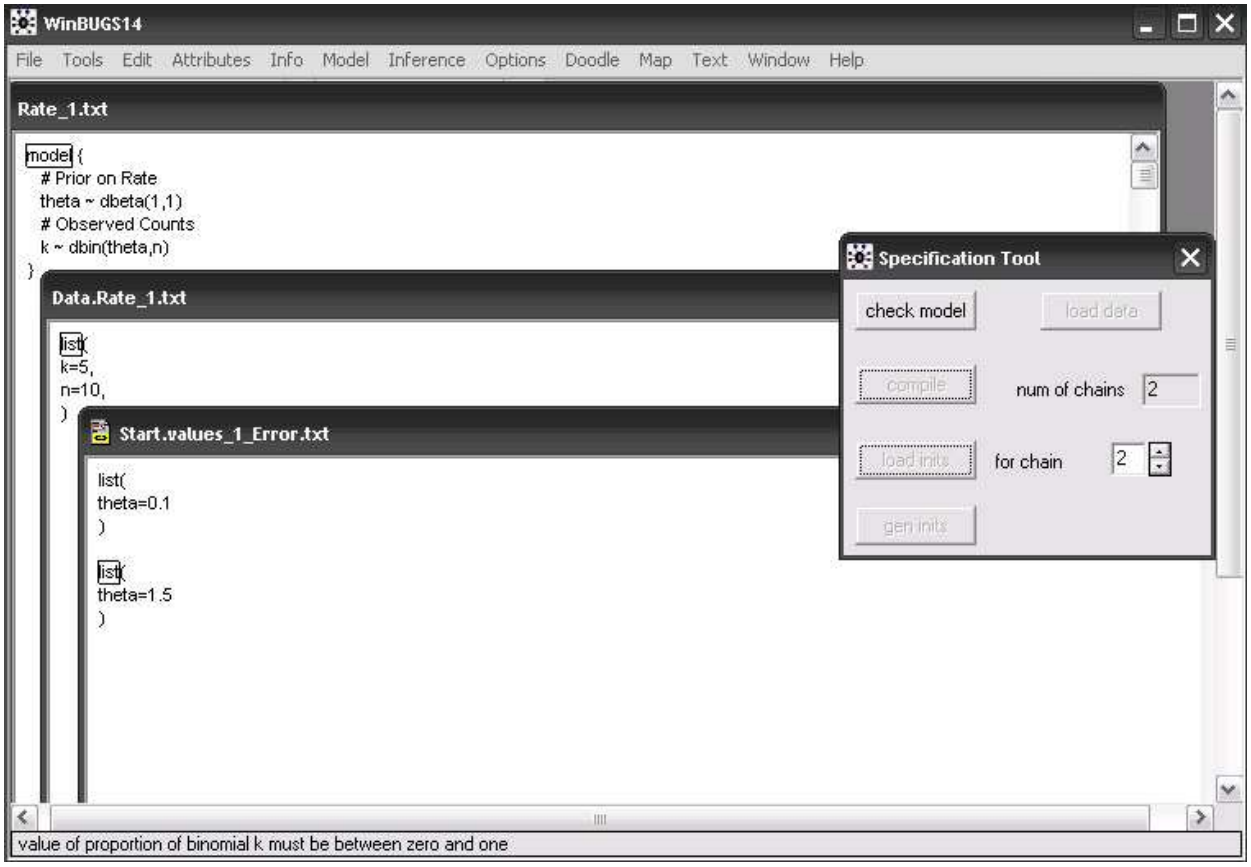


Figure 2.9: WinBUGS error message as a result of an incorrect starting value.

- `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 Divergence Information Criterion (DIC) statistic that the authors of WinBUGS like (because they invented it). The DIC statistic is intended to be used for model selection, but is open to challenge. 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`. You can plot the histogram of these sampled values using Matlab (see `Rate_1.m`). It should look something like the jagged line in Figure 2.10. Because the probability of any value 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,

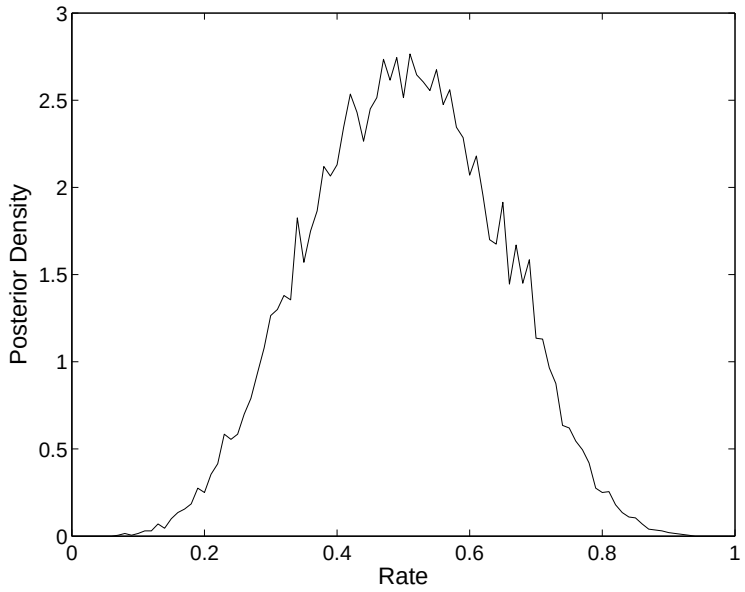


Figure 2.10: Posterior distribution of rate θ for $k = 5$ successes out of $n = 10$ trials, based on 20,000 posterior samples.

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 (1997) 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.

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 a R variable for further analysis. The code we are using to do this is below (see also `Rate_1.R`).

```
setwd("D:/WinBUGS_Book/R_codes") #Set the working directory
library(R2WinBUGS) #Load the R2WinBUGS package
bugsdir = "C:/Program Files/WinBUGS14"

k = 5
n = 10

data    = list("k", "n")
myinits = list(
   list(theta = 0.1),
   list(theta = 0.9))
```

```
parameters = c("theta")

samples = bugs(data, inits=myinits, parameters,
               model.file ="Rate_1.txt",
               n.chains=2, n.iter=20000, n.burnin=0, n.thin=1,
               DIC=F, bugs.directory=bugsdir,
               codaPkg=F, debug=T)
```

Some of these 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` is set to FALSE, WinBUGS returns a variable that contains the results of the sampling run. If `codaPkg` is set to TRUE, WinBUGS returns a variable that contains the file names of the 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 recorded 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 DIC statistic. 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.

WinBUGS returns the sampled values of θ in the R variable `samples`. You can access these values by typing `samples$sims.array`. You can also plot the histogram of these sampled values using R (see `Rate_1.R`). Besides the sequence of posterior samples, WinBUGS also returns some useful statistics to R. You can access the summary statistics of the posterior samples, as well as the $\hat{R}$ statistic mentioned in the previous section by typing `samples`.

2.3 Online Help and Useful URLs

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.

For Mac users

You can run WinBUGS on Macs using emulators, such as Darwine. As best we know, you need a Dual Core Intel based Mac and the latest stable version of Darwine to be able to use R2WinBUGS.

- The Darwine emulator is available at `http://www.kronenberg.org/darwine/`.

- The R2WinBUGS reference manual on the R-project webpage (`http://cran.r-project.org/web/packages/R2WinBUGS/index.html`) provides instructions on how to run R2winBUGS on Macs.

- Further information on how to run R2WinBUGS on Macs is available at `http://ggorjan.blogspot.com/2008/10/runnning-r2winbugs-on-mac.html` and `http://idiom.ucsd.edu/~rlevy/winbugsonmacosx.pdf`.

- Further information on how to run WinBUGS on Macs using a Matlab or R interface is available at `http://web.mit.edu/yarden/www/bayes.html` and `http://www.ruudwetzels.com/index.php?src=MacBUGS`.

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 (`http://www.mrc-bsu.cam.ac.uk/bugs/faqs/contents.shtml`) and how to run WinBUGS using a Matlab interface (`http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/remote14.shtml`).

- The R2WinBUGS reference manual on the R-project webpage (`http://cran.r-project.org/web/packages/R2WinBUGS/index.html`) provides instructions on how to run R2winBUGS under Linux.

Further Reading

Ntzoufras, I. (2009). *Bayesian modeling using WinBUGS*. Hoboken, NJ: Wiley.

Ntzoufras (2009) provides an easily accessible introduction to the use of WinBUGS. The book also presents a variety of Bayesian modeling examples, with the emphasis on Generalized Linear Models.

Spiegelhalter, D., Best, N. & Lunn, D. (2003). *WinBUGS User Manual 1.4*. MRC Biostatistic Unit, Cambridge, UK.

The WinBUGS User Manual provides an introduction to the use of WinBUGS, including a useful tutorial and various tips and tricks for new users.

CHAPTER 3

INFERENCES INVOLVING BINOMIAL DISTRIBUTIONS

3.1 The Difference Between Two Rates

Following up the binomial example in the previous chapter, suppose that now 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 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.

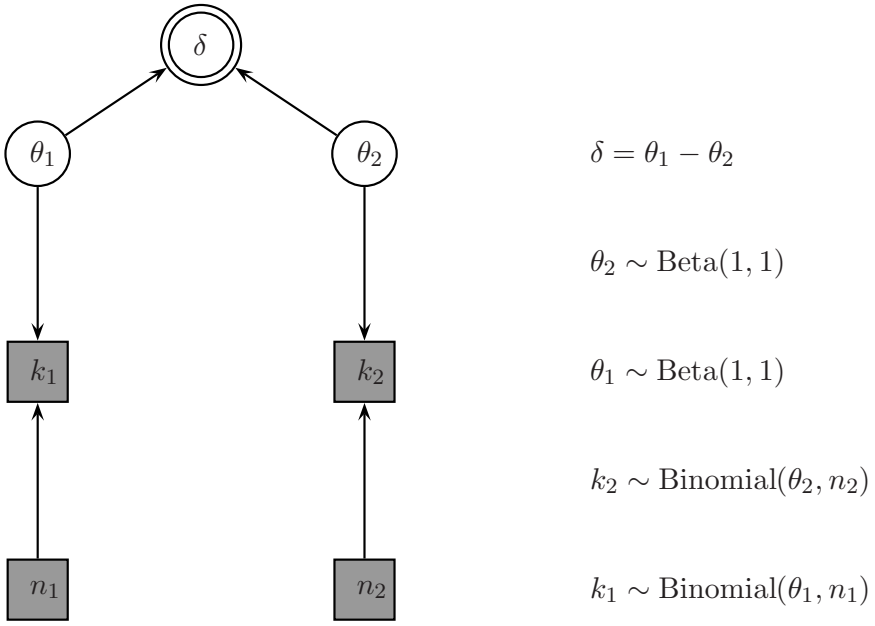


Figure 3.1: Graphical model for inferring the difference in the rates of two binary process.

The graphical model representation for this problem is shown in Figure 3.1. The new notation is that the deterministic variable δ is shown by a double-bordered node.

The following code implements the graphical model in WinBUGS.

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

The code `Rate_2.m` (Matlab) or `Rate_2.R` (R) makes up some data (i.e., sets values $k_1 = 5$, $k_2 = 7$,

$n_1 = n_2 = 10$), and then calls WinBUGS to sample from the graphical model. WinBUGS returns to Matlab/R 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 3.2.

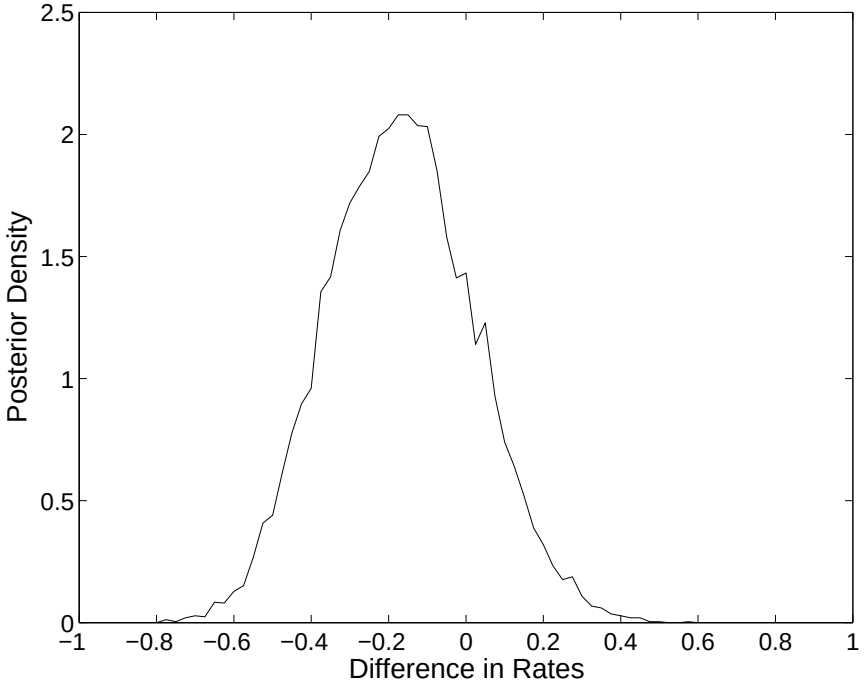


Figure 3.2: Posterior distribution of the difference between two rates $\delta = \theta_1 - \theta_2$.

There are many computations that the posterior samples for δ support that might usefully summarize the full information in the posterior distribution. The code `Rate_2.m` (Matlab) or `Rate_2.R` (R) produces a set of these, including

- The mean value, which approximates the expectation of the posterior. This is the point-estimate corresponding to quadratic loss (i.e., try to pick a single value close to the truth, with bigger deviations from the truth being punished more heavily).
- The value with maximum density in the posterior samples, approximating the posterior mode. This known as the maximum a posteriori (MAP) estimate, and is the same as the maximum likelihood estimate (MLE) for 'flat' priors. This point-estimate corresponds to 0-1 loss (i.e., try to pick the single most likely value). Estimating the mode requires 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 point-estimate corresponding to linear loss.
- The 95% credible interval. This is much like a confidence interval in standard statistics, giving 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. It is easy to see how to modify the Matlab/R code to produce credible intervals for criteria other than 95%.

For the current problem, the mean of δ is -0.17, the mode is -0.20, the median is -0.17, and the 95% credible interval is $[-0.52, 0.21]$.

Exercises

Once the code is working, here are some exercises.

1. Compare the data sets $k_1 = 8$, $n_1 = 10$, $k_2 = 7$, $n_2 = 10$ and $k_1 = 80$, $n_1 = 100$, $k_2 = 70$, $n_2 = 100$.
2. Try the data $k_1 = 0$, $n_1 = 1$, $k_2 = 0$, $n_2 = 5$.
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?

3.2 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 3.3.

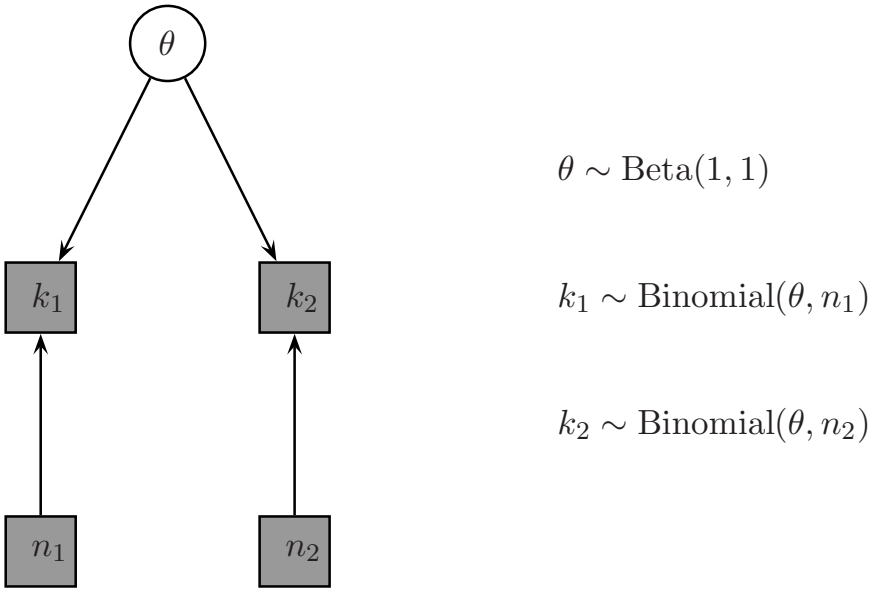


Figure 3.3: Graphical model for inferring the common rate underlying two binary processes.

An equivalent graphical model, using plate notation, is shown in Figure 3.4. Plates are bounding rectangles that enclose independent replications of graphical structure within the 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 computer programming languages (including WinBUGS itself).

The following code implements the graphical model in WinBUGS.

```
# Inferring A Common Rate
model{
   # Prior on Single Rate
   theta ~ dbeta(1,1)
   # Observed Counts
   k1 ~ dbin(theta,n1)
   k2 ~ dbin(theta,n2)
}
```

The code `Rate_3.m` (Matlab) or `Rate_3.R` (R) makes up some data (i.e., set values for k_1, k_2, n_1 and n_2), and then call WinBUGS to sample from the graphical model.

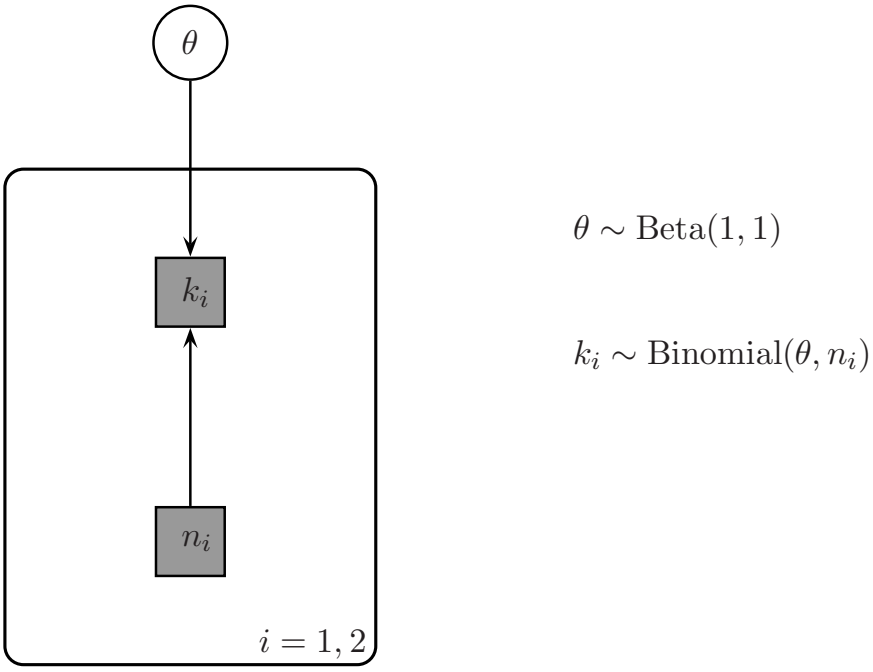


Figure 3.4: Graphical model for inferring the common rate underlying two binary processes, using plate notation

Exercises

Once the code is working, here are some exercises.

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?
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?
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.
4. Try the data $k_1 = 0$, $n_1 = 1$, $k_2 = 2$, $n_2 = 100$.

3.3 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 measurement of data. The most useful part of this bridge is that data allows 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 things we can think about, and they are all important and useful.

- First, the *prior* over parameters captures our initial assumptions or state of knowledge about the psychological variables they represent.
- Secondly, the *prior predictive* tells us what data to expect, given our model (this is the combined likelihood function defined by the sampling distributions and graph structure of a graphical model)

and our current state of knowledge (this is what is represented by the priors). 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* 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* tells us what data 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. It can be viewed as a way to assess the 'adequacy' or 'fit' of a model, because it shows whether the model is able to describe the patterns in the data it was designed to explain.

As an example to illustrate these ideas, we stay with a common rate problem, but generalize it to more than two observed processes. Say, for example, a set of $i = 1, \ldots, m$ people each provide a number of successes k_i out of n binary trials, and everybody is assumed to have the same underlying rate of success θ. The graphical model for this is shown in Figure 3.5.

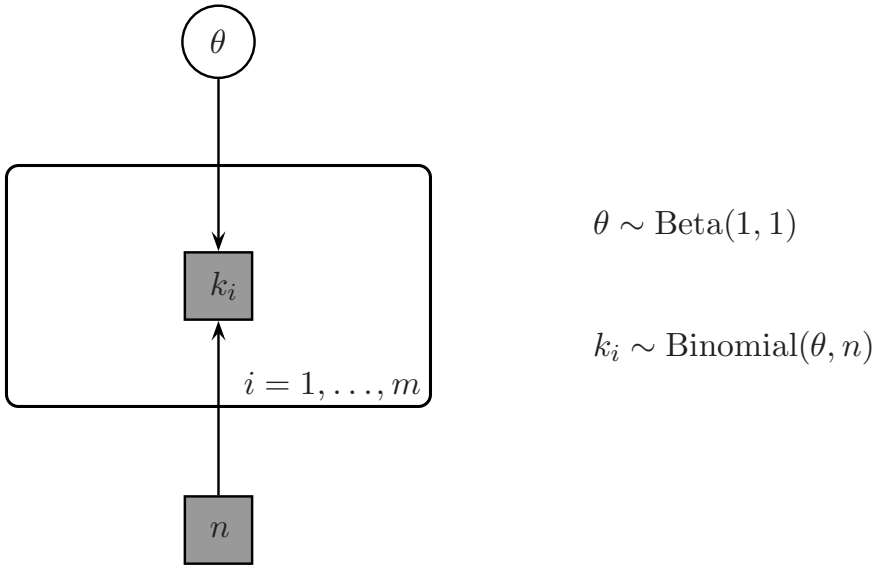


Figure 3.5: Graphical model for the common rate underling many binary processes.

The following code implements the graphical model in WinBUGS, and provides sampling for not just the posterior, but also for the prior, prior predictive and posterior predictive. 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 likelihood, but relying on the prior rate.

Posterior predictive sampling is achieved by the variable `postpredk` that samples data using the same Binomial likelihood, but relying on a 'cut' copy of the posterior rate. WinBUGS provides the cut functionality for making inferences about missing or unobserved data, and the posterior predictive can be conceived in that way, as the unavailable next data sample that would be observed. The key feature of cutting a variable is that sampling can flow downwards, but inference cannot flow upwards. This means that inferences about θ are not affected by considering additional missing posterior predictive data.

```
# Prior And Posterior Prediction
model{
   # Observed Counts
```

```
   for (i in 1:m){
      k[i] ~ dbin(theta,n)
   }
   # Prior on Common Rate
   theta ~ dbeta(1,1)
   # Prior Predictive
   # This Is A "Dummy" Variable That Copies The Prior
   # But Is Never Updated By Data
   thetaprior ~ dbeta(1,1)
   priorpredk ~ dbin(thetaprior,n)
   # Posterior Predictive
   # The Cut Function Allows Sampling To Flow Forward
   # But Prevents Inference Flowing Back
   theta.cut <- cut(theta)
   postpredk ~ dbin(theta.cut,n)
}
```

The code `Rate_4.m` (Matlab) or `Rate_4.R` (R) makes up some data, 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 3.6.

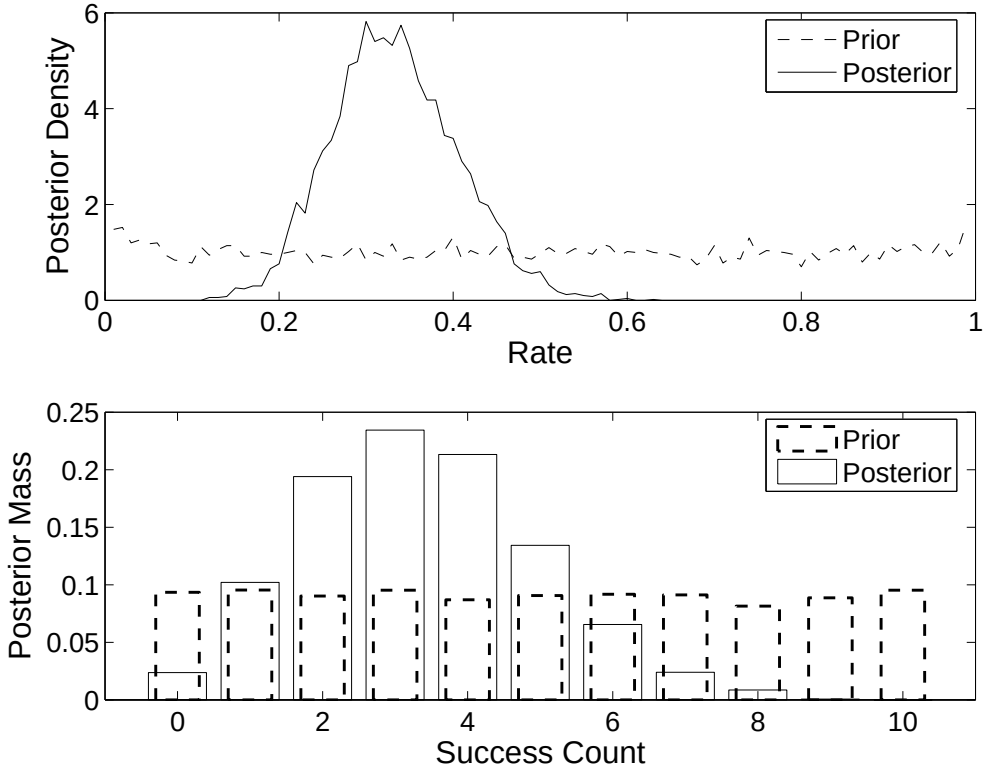


Figure 3.6: Prior and posterior for the success rate (top panel), and prior and posterior predictive for counts of the number of successes (bottom panel), based on data giving $k[i] = \{3, 1, 4, 5\}$ successes out of $n = 10$ trials.

Exercises

1. Make really sure you understand the prior, posterior, prior predictive and posterior predictive quantities, and how they relate to each other (e.g., why is the top panel of Figure 3.6 a line plot, while the bottom panel is a bar graph?). Understanding these ideas is the key to understanding Bayesian analysis. Check your understanding by trying other data sets.
2. Recall the inference about θ for the previous exercise with just two binary processes, with data $k_1 = 0$, $n_1 = 10$, $k_2 = 10$, $n_2 = 10$. How could predictive distributions help understand and diagnose the counter-intuitive aspects of this inference?
3. In October 2009, the Dutch newspaper "Trouw" reported on research conducted by Hester Trompetter, a student from the Radboud University in the city of Nijmegen. For her undergraduate thesis, Hester 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. Hester confidently rejected the suggestion that her study may have been too small to draw reliable conclusions: "If I would have talked to more people, the result would have changed by one or two percent at the most." Is Hester correct? Use WinBUGS to find the 95% credible interval for the expected number of bullied residents, assuming you would test a new group of 121 older adults living in nursing homes. Do you think this credible interval is plausible, or is it still too large or too small?

3.4 Distributions, Information and Updating

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 "prior successes" and "prior failures", respectively. This means, 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 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. The catch-cry is that "today's posteriors are tomorrow's priors."

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 posterior to the prior, they are said to have a "conjugate" relationship. This is valued a lot in analytic approaches to Bayesian inference, because it makes for tractable calculations. It is not so important for that reason in computational approaches, because sampling methods can handle easily 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.

This example gives a little more detail on the interpretation of counts for rate problems. The graphical model for the problem is shown in Figure 3.7. It is essentially the same simple model used in Chapter 2, except the prior on the rate is now $\theta \sim \text{Beta}(\epsilon, \epsilon)$, where $\epsilon > 0$ is intended to be a small number.

The prior distribution $\theta \sim \text{Beta}(0, 0)$ is known as the "Haldane" prior. It expresses a certain type of ignorance about a rate parameters, as neatly explained by Jaynes (2003, pp. 382–385). Through the interpretation of prior counts, a $\text{Beta}(0, 0)$ corresponds to having seen nothing about the binary process. The widely-used uniform prior $\text{Beta}(1, 1)$, however, corresponds to having seen one success and one failure. So, the uniform prior makes sense in cases where it is known the binary process can produce both successes and failures (the prior says both have been seen), and the Haldane prior makes sense in cases where not even that information is known. Note that if you start with a Haldane prior, then see a success and a failure, you end up with the uniform prior.

The Haldane prior is improper (i.e., it is not really a distribution, but the limit of a sequence of distributions), but WinBUGS requires proper distributions. This is why the Haldane is approximated using the small constant ϵ in Figure 3.7. The following code implements the graphical model in WinBUGS, using a concrete value $\epsilon = 0.01$. It also allows for the prior distribution to be sampled.

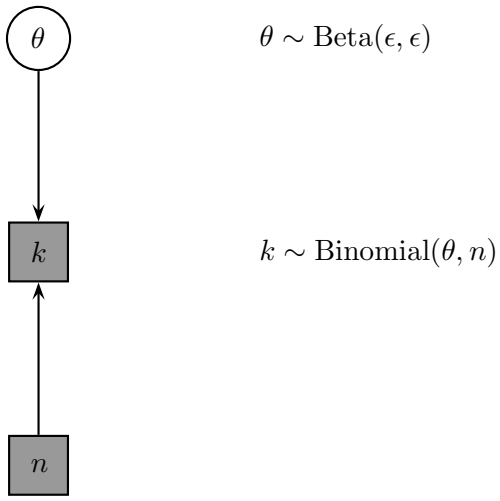


Figure 3.7: Graphical model for inferring the rate of a binary process, using a near-Haldane prior on the rate.

```
# Summarizing Distributions
model {
   # Prior on Rate
   theta ~ dbeta(.01,.01)
   # Observed Counts
   k ~ dbin(theta,n)
   # This Is A "Dummy" Variable That Copies The Prior
   # But Is Never Updated By Data
   thetaprior ~ dbeta(.01,.01)
}
```

The code `Rate_5.m` (Matlab) or `Rate_5.R` (R) sets the data to be one success and one failure (i.e., $k = 1$ from $n = 2$), and then calls WinBUGS to sample from the graphical model. The results are shown in Figure 3.8. The posterior distribution, as should be expected, is essentially uniform over the whole range of possible rates. The prior has almost all of its mass near the extreme rates of 0 and 1.

After a bit of thought, the strange shape of the prior makes sense. When we do not know that both success and failure are possible, it is usually the case that something always succeeds or always fails (i.e., always takes one value or the other). Imagine a whole set of binary processes that involve asking a person repeatedly the Dutch word for "cheese". The binary process is whether or not they give the right answer ("kaas") on each repetition. You would expect that many of these binary processes will have a success rate of 0 (i.e., those people who do not speak Dutch, and guess wrongly on each repetition). You would expect many other of these binary processes to have a success rate of 1 (i.e., those people who do speak Dutch, and keep saying the right answer). Only for a select few of the binary processes would you expect both successes and failures over repetitions (i.e., those people who are learning Dutch, and sometimes remember the word, but sometimes do not). Over all people, the distribution of prior expectations about success rates would look qualitatively like the prior in Figure 3.8 (i.e., lots of non-Dutch speakers, lots of Dutch speakers, and few who are in the learning process).

This problem is a good example of the basic process of Bayesian inference for a model. Start with a prior distribution over the variables that captures—as well as possible—what is known about the relatively likelihoods of each possible value of those variables. Then, as additional information (often in the form of empirical data) become available, update those distributions to incorporate the new knowledge, and (usually) become less uncertain about the values of the variables.

Of course, specifying priors may be easy or hard, in much the same way that specifying the probabilistic

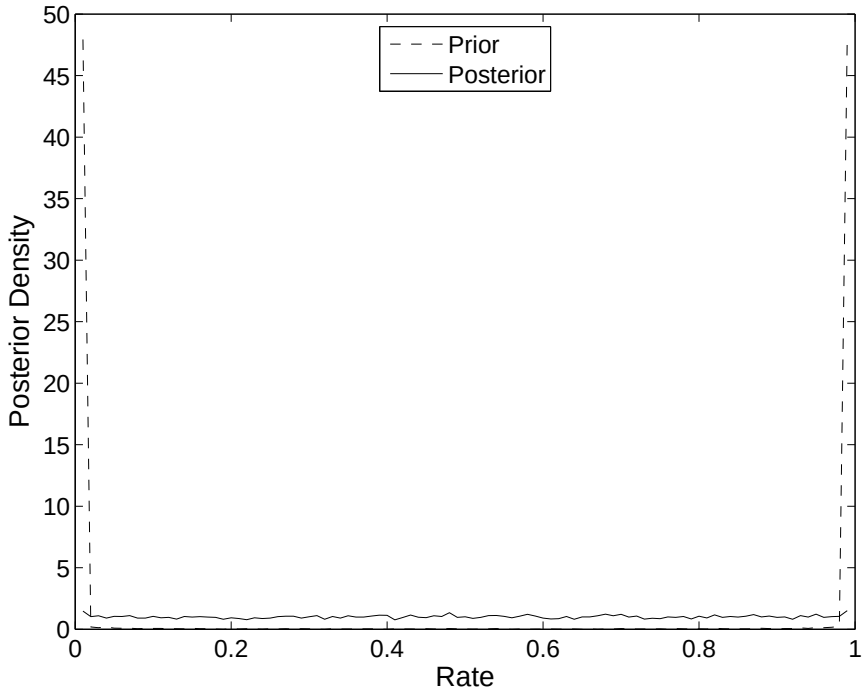


Figure 3.8: Prior and posterior distribution for a rate with a near-Haldane prior, and one observed success and failure.

model (which variables are relevant, how they depend on each other, how they are distributed) may be easy or hard. In fact, the prior distribution really is part of the modeling exercise, because it is the priors together with the likelihood defined by the graph structure and distributions that determine the prior predictive; and these are the data patterns the model as a whole expects to see. One of the attractive features of the Bayesian approach is that all of these prior assumptions are explicitly built into modeling, and, from there, the machinery of Bayesian inference—with its axiomatic grounding in the laws of probability—can then automatically incorporate new information.

Exercises

1. Suppose you are asked to give a point estimate of the prior and posterior distributions in Figure 3.8. How would you answer?

3.5 Summarizing 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 has meant that our parameter space has been one-dimensional. Everything learned from data is incorporated into a single probability distribution representing the relative likelihood 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. This final binomial process example makes both the rate θ *and* the total number n unknown, and so the problem is to infer both simultaneously from counts of successes k.

To make the problem concrete, suppose you have five helpers distribute a bundle of surveys to houses. You know that each bundle contained the same number of surveys, n, but you do not know what that

number was. The only information you have is that the maximum bundle is (say) $N_{\max} = 500$, and so n must be between 1 and $N_{\max}$.

You also do not know what the rate of return for the surveys is. But you are willing to assume that each helper distributed to houses selected in a random enough way that it is reasonable to believe the return rates are the same. Suppose you set a prior on this rate $\theta \sim \text{Uniform}\big(0, 1\big)$.

Inferences can simultaneously be made about n and θ from the actual number of surveys returned for each of the helpers. Assuming the surveys themselves are able to 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.

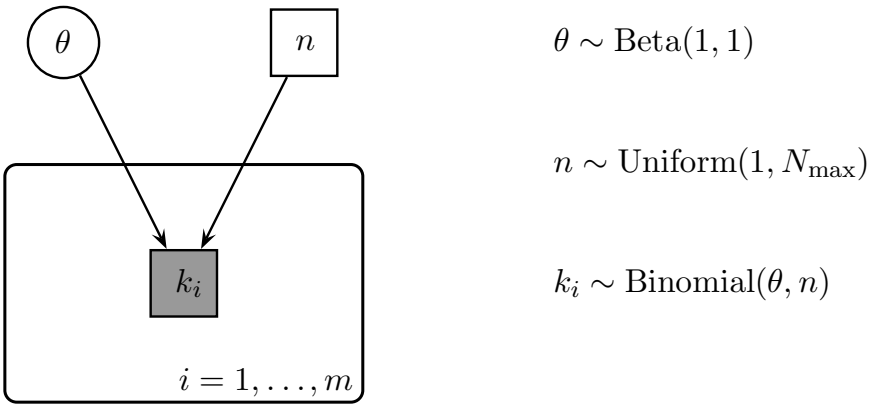


Figure 3.9: Graphical model for the joint inference of n and θ from a set of m observed counts of successes $k_1, \ldots, k_m$.

The graphical model for this problem is shown in Figure 3.9, and the following code implements the graphical model in WinBUGS.

```
# Inferring return rate and numbers of surveys from observed returns
model {
   # Likelihood
   for (i in 1:m){
      k[i] ~ dbin(theta,n)
   }
   # Priors
   theta ~ dbeta(1,1)
   n ~ dunif(0,Nmax)
}
```

The code `Rate_6.m` (Matlab, requires the `Statistics Toolbox`) or `Rate_6.R` (R) uses the data $k = \{16, 18, 22, 25, 27\}$, and then calls WinBUGS to sample from the graphical model. Figure 3.10 shows the joint posterior distribution over n and θ as a scatterplot, and the marginal distributions of each as histograms.

It is clear that the joint posterior distributions 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, and then sampling one from the histogram for θ, and plotting the two-dimensional point corresponding to these samples. Then imagine repeating this process many times. It should be clear the resulting scatterplot would be different from the joint posterior scatterplot in Figure 3.10. So, the joint carries information not available from the marginals.

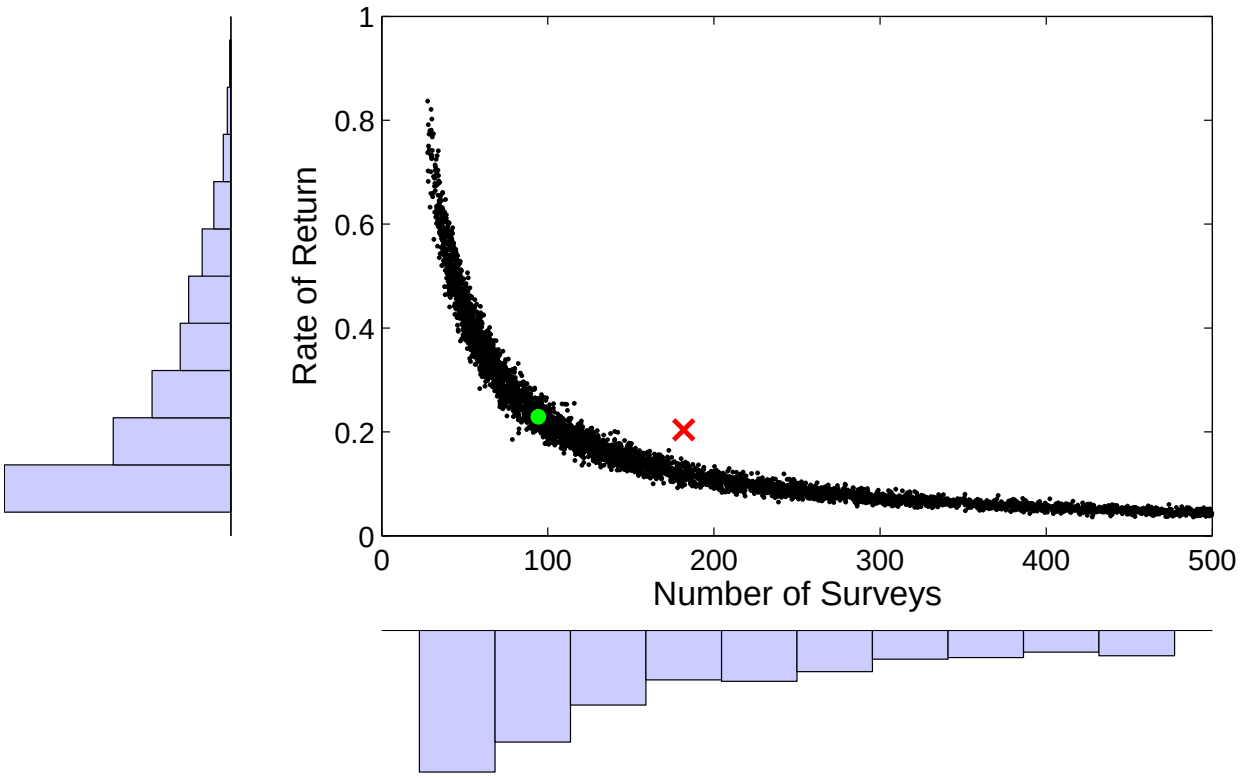


Figure 3.10: Joint posterior distribution (scatterplot) of the probability of return θ and the number of surveys m for observed counts $k = \{16, 18, 22, 25, 27\}$. The histograms show the marginal densities. The red cross shows the expected value of the joint posterior, and the green circle shows the mode (i.e., maximum likelihood), both estimated from the posterior samples.

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 3.10.

Exercises

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 red cross in Figure 3.10 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?

2. The green circle in Figure 3.8 shows an approximation to the mode (i.e., the sample with maximum likelihood) from the joint posterior samples. Does this make sense?

3. Try the very slightly changed data $k = \{16, 18, 22, 25, 28\}$. How does this change the joint posterior, the marginal posteriors, the expected point, and the maximum likelihood point? If you were comfortable with the mode, are you still comfortable? [This example is based heavily on one I read in

a book, but have lost the reference. If you know which one, could you please let me know, so I can acknowledge it? Ta.]

CHAPTER 4

INFERENCES INVOLVING GAUSSIAN DISTRIBUTIONS

4.1 Inferring Means and Standard Deviations

One of the most common inferential problems involving assuming data following a Gaussian (also known as the 'Normal', 'Central', '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 $\lambda = 1/\sigma^2$.

The prior used for μ is intended to be 'flat' and uninformative. 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 seems relatively uncontroversial in the literature.

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.

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

The following code 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_1.m` (Matlab) or `Gaussian_1.R` (R) creates some artificial data, and applies the graphical model to do inference.

Exercises

Once the code is working, here are some exercises.

1. Try a few data sets, varying what you expect the mean and standard deviation to be, and how many data you observe.
2. Try changing the assumption about the upper bound of 10 on the prior for the standard deviation. Characterize those cases where it makes no practical difference to the inferences drawn, and those cases where it does matter. Interpret the difference between these two cases in terms of whether and how the prior assumptions contribute substantial information to the problem.
3. In Matlab or R, plot the *joint* posterior of μ and σ (e.g., using the command `scatterhist` in Matlab, or `plot` in R). Interpret the plot.
4. 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/R code to do this. What does the revised graphical model look like?
5. 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 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/R code to do this. Once again, what does the revised graphical model look like?

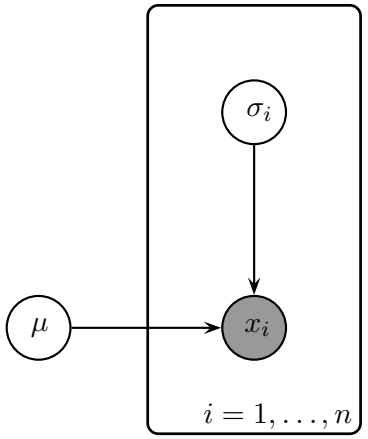
4.2 The Seven Scientists

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 (good) 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, it is just the standard deviation that differs.

Notice the different approach to setting priors about the standard deviations used in this example. This approach has a theoretical basis in scale invariance arguments (i.e., choosing to set a prior so that changing the measurement scale of the data does not affect inference). While the idea is attractive, it quickly gets complicated, and most likely for fundamental and important reasons. The invariant prior turns



$$\mu \sim \text{Gaussian}(0, 0.001)$$

$$\sigma \sim \text{InvSqrtGamma}(0.001, 0.001)$$

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

Figure 4.2: Graphical model for the seven scientists problem.

out (see Jaynes, 2003) to be improper, meaning it is not really a distribution, but the limit of a sequence of distributions, much like the Haldane distribution in the earlier rate problem. Once again, WinBUGS requires proper distributions always be used, and so the InvSqrtGamma$\big(.001, .001\big)$ is intended as a proper approximation to the theoretical improper prior. This raises the issue of whether inference is sensitive to the essentially arbitrary value 0.001. Gelman (2006) raises some other challenges to this approach, and it is probably fair to say it is declining in popularity in the applied literature. But, it is still worth knowing about.

Anyway, the following code implements the graphical model in Figure 4.2 in WinBUGS. Notice that the Inverse-SquareRoot-Gamma prior distribution is implemented by first setting a prior for the precision, $\lambda \sim \text{Gamma}\big(.001, .001\big)$ and then re-parameterization to the standard deviation.

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

The code `Gaussian_2.m` (Matlab) or `Gaussian_2.R` (R) applies the seven scientist data to the graphical model.

Exercises

1. Draw posterior samples using the Matlab/R code, and reach conclusions about the value of the measured quantity, and about the accuracies of the seven scientists.
2. Change the graphical model in Figure 4.2 to use a uniform prior over the standard deviation, as was done in Figure 4.1. Experiment with the effect the upper bound of this uniform prior has on inference. This may be a case—involving many standard deviations, but only getting one datum relevant to each—where the scale invariance ideas might be important.

4.3 Repeated Measurement of IQ

In this example, we will 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 Gaussian error with zero mean, but some unknown precision. The mean of the Gaussian a person's test scores corresponds to their IQ measure. 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 attempts to set priors (these are re-visited 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.

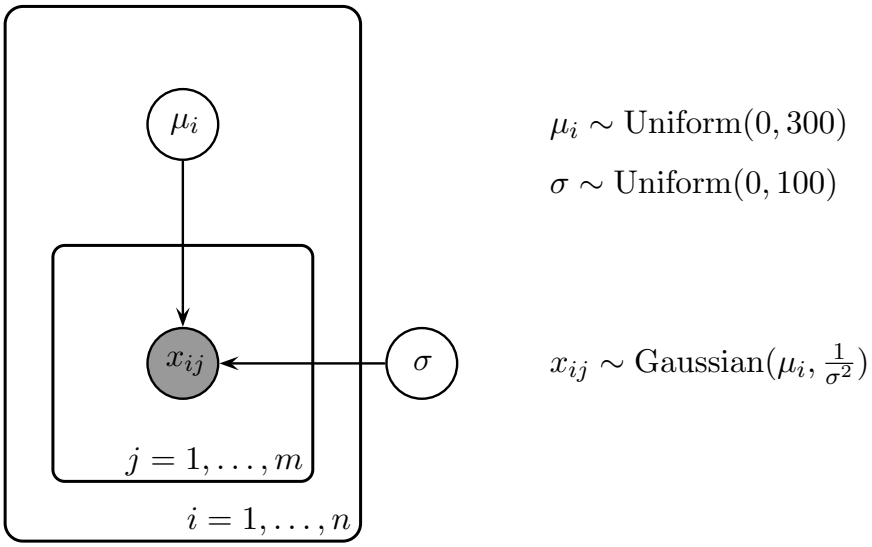


Figure 4.3: Graphical model for inferring the IQ from repeated measures.

The following code implements the graphical model in WinBUGS.

```
# Repeated Measures Of IQ
model{
   # Data Come From A Gaussian 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 `Gaussian_3.m` (Matlab) or `Gaussian_3.R` (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

1. Draw posterior samples using the Matlab/R Code. Estimate each person's IQ as the mean of the posterior distribution for their μ_i. What can we say about the precision of the three IQ tests?
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] ~ dnorm(100,.0044)`, instead of `mu[i] ~ dunif(0,300)`. Make this change in the WinBUGS script, and re-run the inference. How do the estimates of IQ given by the means change? Why?
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?

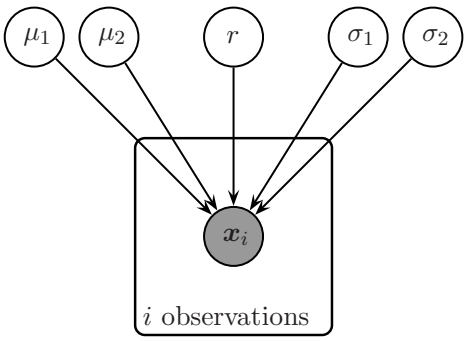
CHAPTER 5

SOME EXAMPLES OF BASIC DATA ANALYSIS

5.1 Pearson Correlation

The Pearson-product moment correlation coefficient, usually denoted r, is a very widely-used measure of the relationship between two variables. It ranges between +1, indicating a perfect positive linear relationship, to 0, indicating no linear relationship, to −1 indicating a perfect negative relationship. Usually the correlation r is reported as a single number (a 'point estimate'), perhaps together with a frequentist significance test.

But, rather than just having a single number to measure the correlation, it seems like 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. These approaches might have some merit, but they also clearly have problems. And, it is easy to set up a graphical model that allows inferences about the correlation coefficient for any data and set of prior assumptions about the correlation.



$$\mu_1, \mu_2 \sim \text{Gaussian}(0, 0.001)$$

$$\sigma_1, \sigma_2 \sim \text{InvSrqtGamma}(0.001, 0.001)$$

$$r \sim \text{Uniform}(-1, 1)$$

$$\boldsymbol{x}_i \sim \text{MultivariateGaussian}\left((\mu_1, \mu_2) , \begin{bmatrix} \sigma_1^2 & r\sigma_1\sigma_2 \\ r\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}^{-1} \right)$$

Figure 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 person or unit observed, 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 and standard deviations of the two dimensions, and the correlation coefficient that links them.

In Figure 5.1, the variances are given the approximations to theoretically 'non-informative' priors, as discussed earlier. The correlation coefficient itself is given a uniform prior over its possible range. All of these choices would be easily modified, with one obvious change being to give the correlation prior more density around 0. One of the best features of the computational approach to Bayesian inference we are using is that it is quick and easy to try alternative assumptions like these.

The following code implements the graphical model shown in Figure 5.1 in WinBUGS.

```
# Pearson Correlation
model {
   # Likelihood
   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` (Matlab) or `Correlation_1.R` (R) includes several data sets, described in the Exercises below, and uses WinBUGS to sample from the graphical model.

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

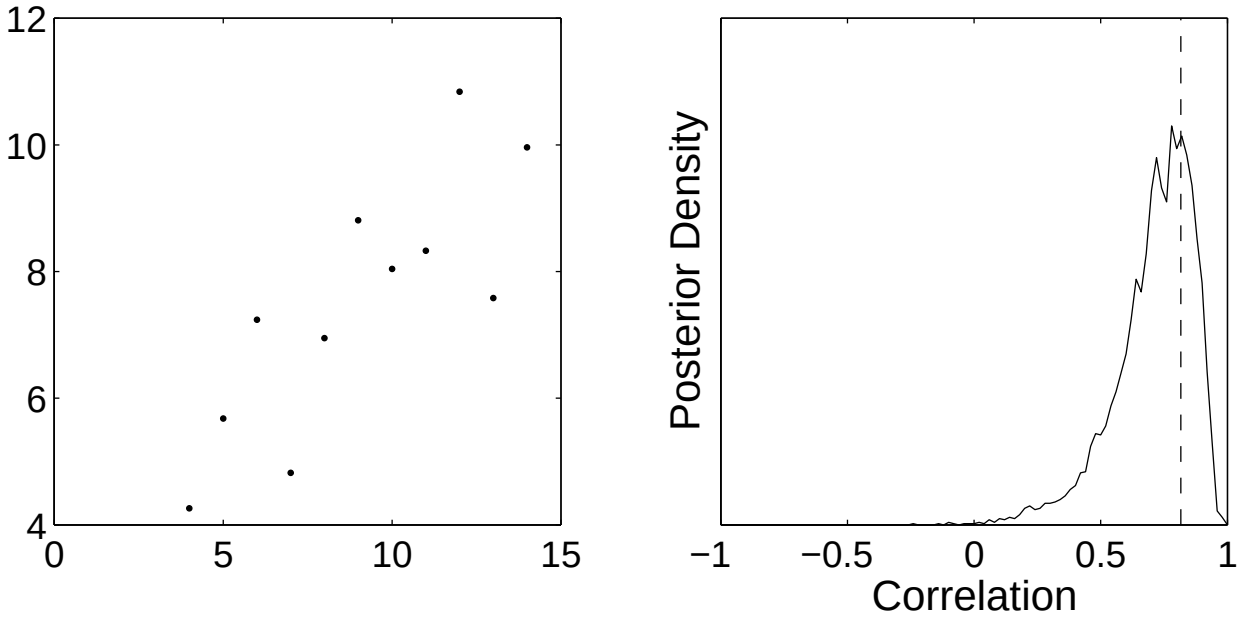


Figure 5.2: Data (left panel) and posterior distribution for correlation coefficient (right panel). The broken line shows the frequentist point-estimate.

Exercises

1. The second data set in the Matlab/R code is just the first data set from Figure 5.2 repeated twice. Interpret the differences in the posterior distributions for r for these two data sets.

2. Compare the scatterplots and posterior distributions for the first data set to those from the third, fourth and fifth data sets in the Matlab/R code. What is the moral of the story?

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 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 D 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, for a Bayesian approach). 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

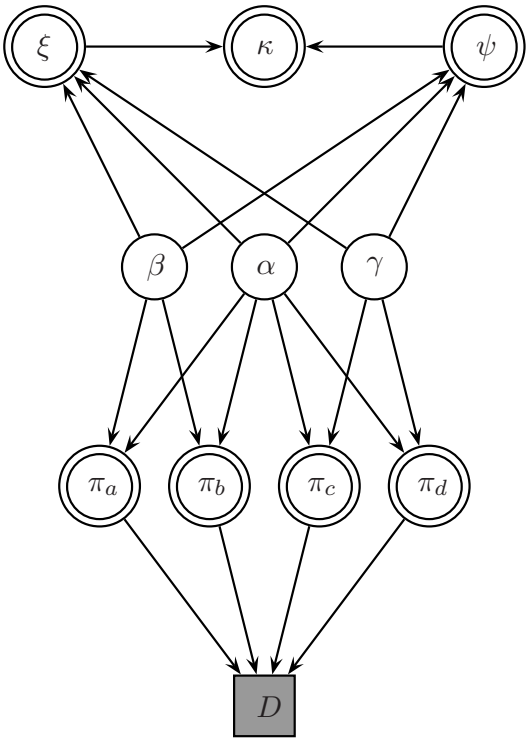
$$\kappa = \frac{p_o - p_e}{1 - p_e}.$$

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.

The graphical model for a Bayesian version of kappa is shown in Figure 5.3. The key latent variables are α, β and γ: Once you understand these, the result of the model is easier to follow.

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 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', $\pi_d = (1 - \alpha)\gamma$.



$$\kappa = (\xi - \psi) / (1 - \psi)$$
$$\xi = \alpha\beta + (1 - \alpha)\gamma$$
$$\psi = (\pi_a + \pi_b)(\pi_a + \pi_c) + (\pi_b + \pi_d)(\pi_c + \pi_d)$$

$$\alpha \sim \text{Beta}(1, 1)$$
$$\beta \sim \text{Beta}(1, 1)$$
$$\gamma \sim \text{Beta}(1, 1)$$

$$\pi_a = \alpha\beta$$
$$\pi_b = \alpha(1 - \beta)$$
$$\pi_c = (1 - \alpha)(1 - \gamma)$$
$$\pi_d = (1 - \alpha)\gamma$$

$$D \sim \text{Multinomial}([\pi_a, \pi_b, \pi_c, \pi_d], n)$$

Figure 5.3: Graphical model for inferring the kappa coefficient of agreement.

These probabilities, in turn, describe how the observed data, D, 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 D allows inferences to be made about the key rates α, β and γ. The remaining variables in the graphical model in Figure 5.3 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 observed 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 (prize to the first person to do this correctly). Finally κ is the chance corrected measure of agreement on the -1 to $+1$ scale, given by $\kappa = (\xi - \psi) / (1 - \psi)$.

The following code 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
   d[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_1.m` (Matlab) or `Kappa_1.R` (R) passes several data sets, described in the Exercises below, to WinBUGS to sample from the graphical model.

Exercises

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 respitory 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 $d = 210$.

   Plot 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?

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 of hearing loss in relation to expert assessment. Of those children assessed as having hearing loss by the expert, 20 were correctly identified by the nurse and 7 were missed. Of those assessed as not having hearing loss by the expert, 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, plot 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?

3. *Salem Witch Trial Data* Believe it or not, Mather and Mather (1979) report data on the association between whether or not people were diagnosed as possessed by Satan and whether they were treated by being hanged or not being hanged. Their data are reproduced below.

	Not Possessed	Possessed
Not Hanged	157	0
Hanged	13	0

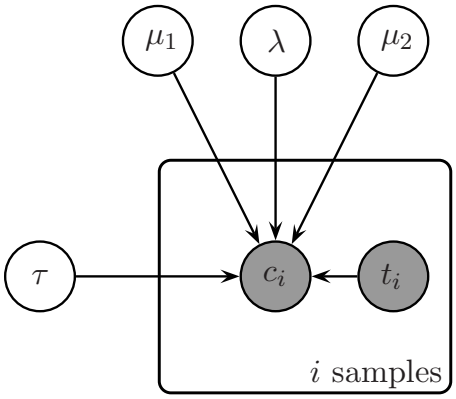
   Apply the kappa graphical model to these data. There are at least two serious points to be made. There are probably many less serious ones. Concentrate on the former. (Hints: What would a frequentist do with the column of zero counts, and does it make a difference which method you treat as the gold standard?)

5.3 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.

I don't have any idea what that means, but it gets up the quinella of sounding neuro and clinical, and so must be impressive and eminently fundable work.

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 fairly 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.



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

Figure 5.4: Graphical model for detecting a single change-point in time series.

Figure 5.4 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, and so 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 uninformative priors are given to the means and the precision.

The following code 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,tmax)
}
```

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

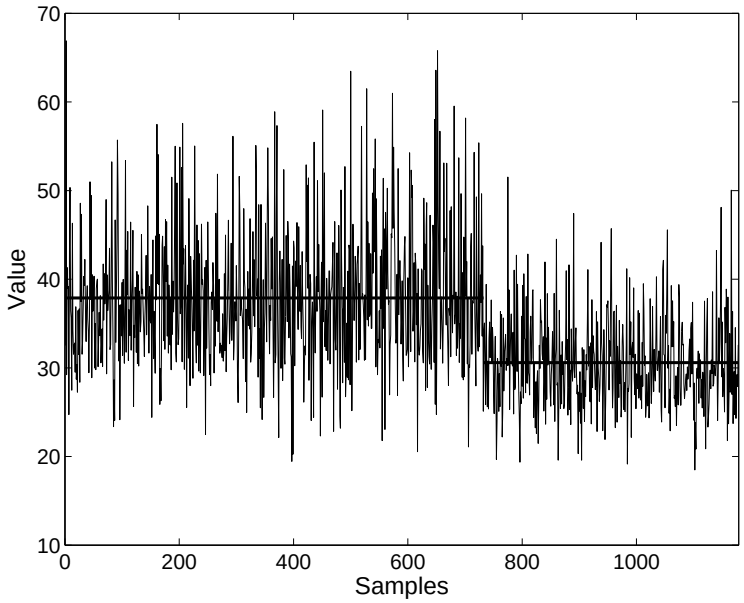


Figure 5.5: Identification of change-point in time series data.

needs to take on values 1 and 2. This is the reason why `z` is transformed to `z1`. Study this code and make sure you understand what the `step` function accomplishes in this example.

The code `ChangeDetection_1.m` (Matlab) or `ChangeDetection_1.R` (R) applies the model to the near-infrared spectrographic data. The 1178 count data did not come with times, so uniform sampling is assumed, giving $t = 1, \ldots, 1778$.

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 should look something like Figure 5.5.

Exercises

1. Draw the posterior distributions for the change-point, the means, and the common standard deviation.
2. Figure 5.5 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?
3. Imagine that you apply this model to a data set that has two change-points instead of one. What could happen?

5.4 Censored Data

Since 13 April 2005, Cha Sa-soon—a 68-year old grandmother living in Jeonju, South Korea—had repeatedly tried to pass the written exam for a driving licence. In South Korea, this exam features 50 four-choice questions; in order to pass, one is required to score at least 60 points out of a maximum of 100. In the following, we assume that each correct answer is worth 2 points, so that in order to pass one needs to answer at least 30 questions correctly.

What makes Cha Sa-soon special is that she failed to pass the test on 949 consecutive occasions, spending the equivalent of 4,200 US dollar on application fees. In her last, 950th attempt, Cha Sa-soon scored the required minimum of 30 correct questions and finally obtained 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 any one question correctly? Note that we assume that each question is equally difficult, and that Cha Sa-soon does not learn from her earlier attempts.

What makes these data special is that for the failed attempts, we do not know the precise scores—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 above. The following code, inspired by Gelman and Hill (2007, p. 405), shows how WinBUGS deals with censored data.

```
model
{
  for (i in 1:nattempts)
  {
    # If the data were unobserved (y[i] equals "cens"), then they must
    # range from 15 to 25:
    z.low[i]  <- 15*equals(y[i],cens)
    z.high[i] <- nquestions - ((nquestions-25)*equals(y[i],cens))

    z[i]       ~ dbin(theta,nquestions)I(z.low[i],z.high[i])
  }
  # Uniform prior on theta:
  theta ~ dbeta(1,1)
}
```

Here the vector `y[i]` contains the data, where a failed attempt is scored as 29 (i.e., the value of `cens`). Vector `z[i]` also contains the data, but here a failed attempt is denoted by `NA`.

Note the use of the `equals` command, a command that returns 1 when its arguments match, and 0 when they mismatch; thus, when `y[i]` is equal to `cens`, `z.low[i]` is 15, and otherwise it is 0. In similar fashion, the line that begins with `z.high[i]` states that when `y[i]` is equal to `cens`, `z.high[i]` is 25, and otherwise `z.high[i]` is `nquestions` (i.e., 50).

The code `Cha_Sa-soon.R` (R) applies the model to the data from Cha Sa-soon. The posterior density for θ is shown in Figure 5.6. As can be seen, the posterior is relatively peaked—despite the fact that we do not know the actual scores for 949 of the 950 results, we were still able to infer a lot about θ.

Exercises

1. Do you think Cha Sa-soon could have passed the test by just guessing? Give at least two arguments to support your claim.
2. What happens when you increase the interval in which you know the data are located? (i.e., the interval that is now 15-25)
3. What happens when you decrease the number of failed attempts?
4. What happens when you increase Cha Sa-soon's final score?

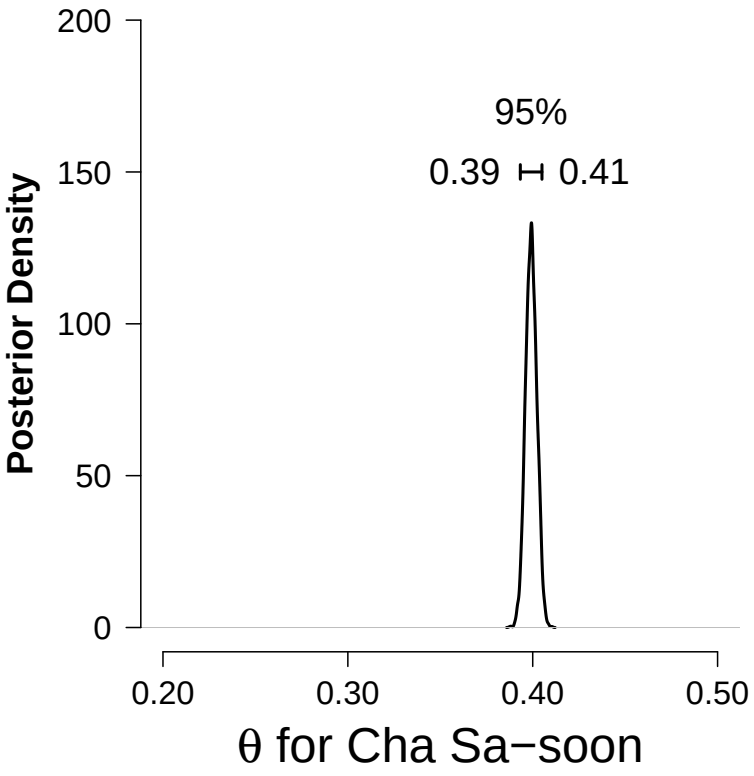


Figure 5.6: Posterior density for Cha Sa-soon's probability of answering any four-choice question correctly.

CHAPTER 6

EXAMS, QUIZZES, LATENT GROUPS, AND MISSING DATA

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 and 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.

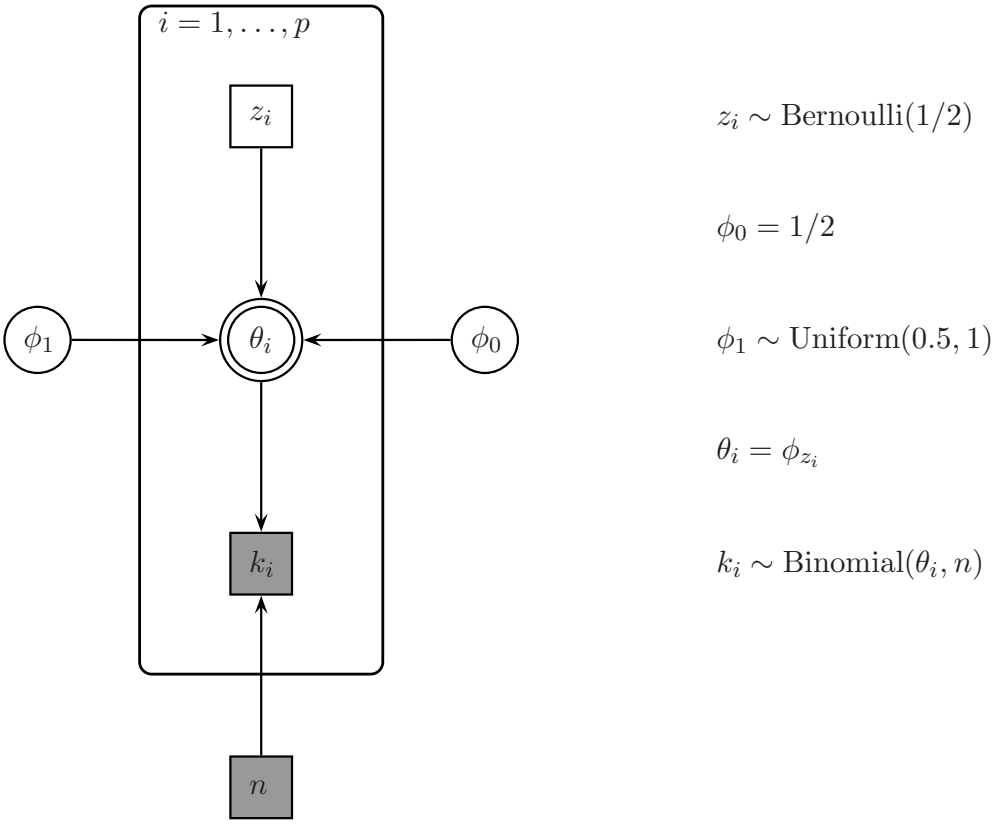


Figure 6.1: Graphical modeling for inferring membership of two latent groups, with different rates of success in answering exam questions.

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 ϕ_0, if the person is in the guessing group, or ϕ_1 if the person is in the knowledge group. Which group they are in is determined by their binary indicator variable z_i, with $z_i = 0$ if the ith person is in the guessing group, and $z_i = 1$ is they are in the knowledge group. This means we can define

$\theta_i = \phi_{z_i}$.

We assume each of these indicator variables 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 $\phi_0 = 1/2$. For the knowledge group, we use a prior where all rate possibilities greater than 1/2 are equally likely, so that $\phi_1 \sim \text{Uniform}\big(0.5, 1\big)$.

The following code implements the graphical model in WinBUGS. Notice the use of a dummy variable `z1[i] <- z[i]+1`, which—just as in the previous change-point example—allows WinBUGS array structures to be indexed in assigning `theta[i]`.

```
# Exam Scores
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
   }
   # First Group Just Guesses
   phi[1] <- 0.5
   # Second Group Has Some Uknown Greater Rate Of Success
   phi[2] ~ dunif(0.5,1)
   # Data Follow Binomial With Rate Given By Each Person's Group Assignment
   for (i in 1:p){
      theta[i] <- phi[z1[i]]
      k[i] ~ dbin(theta[i],n)
   }
}
```

The code `ExamsQuizzes_1.m` (Matlab) or `ExamsQuizzes_1.R` (R) makes inferences about group membership, and the success rate of the knowledge group, using the model.

Exercises

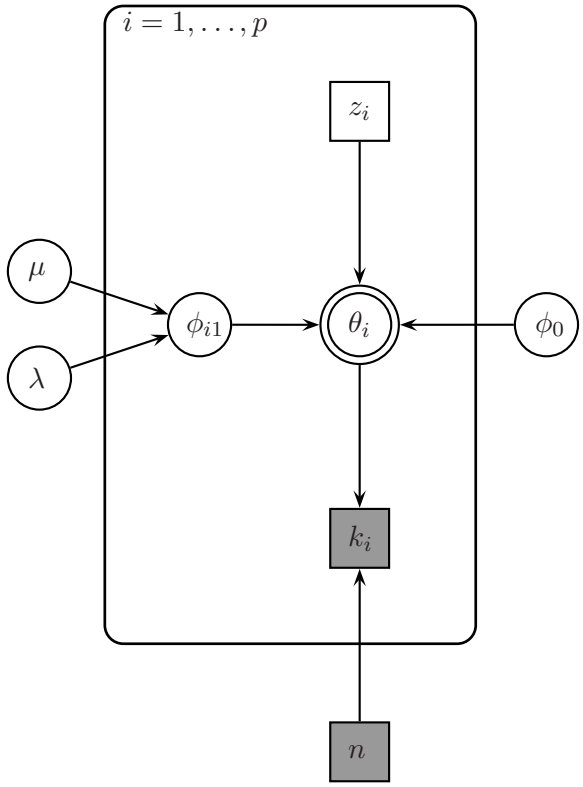
1. Draw some conclusions about the problem from the posterior distribution. Who belongs to what group, and how confident are you?
2. The initial allocations of people to the two groups in this code is random, and so will be different every time you run it. Check that this does not affect the final results from sampling.
3. Include an extra person in the exam, with a score of 28 out of 40. What does their posterior distribution for z tell you?
4. What happens if you change the prior on the success rate of the second group to be uniform over the whole range $(0, 1)$, and so allow for worse-than-guessing performance?
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), $z_i \sim \text{Bernoulli}\big(0.9\big)$?

6.2 Exam Scores With Individual Differences

Probably the best thing about the previous example is that it shows how naturally and easily sampling can find discrete latent groups. But the model itself 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 rich 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 censor the sampled success rates to the valid range.



$$z_i \sim \text{Bernoulli}(1/2)$$

$$\phi_0 = 1/2$$

$$\phi_{i1} \sim \text{Gaussian}(\mu, \lambda)$$

$$\mu \sim \text{Uniform}(0.5, 1)$$

$$\lambda \sim \text{Gamma}(0.001, 0.001)$$

$$\theta_i = \begin{cases} \phi_0 & \text{if } z_i \text{ is 0} \\ \phi_{i1} & \text{if } z_i \text{ is 1} \end{cases}$$

$$k_i \sim \text{Binomial}(\theta_i, n)$$

Figure 6.2: Graphical modeling 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_{i1} for the ith person. These success rates are drawn from a Gaussian distribution with mean μ and precision 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.

The following code implements the graphical model in WinBUGS. Notice that is includes a posterior predictive variable `predphi` for the knowledge group success rates of each person.

```
# Exam Scores With Individual Differences
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
   }
   # The Second Group Now Allows Individual Differences
   # So There Is a Rate Per Person
   for (i in 1:p){
      # First Group Is Still Just Guesses
      theta[i,1] <- 0.5
      # Second Group Drawn From A Censored Gaussian Distribution
```

```
      theta[i,2] ~ dnorm(mu,lambda)I(0,1) # Censor The Probability To (0,1)
   }
   # Second Group Mean, Precision (And Standard Deviation)
   mu ~ dunif(0.5,1) # Greater Than 0.5 Average Success Rate
   lambda ~ dgamma(.001,.001)
   sigma <- 1/sqrt(lambda)
   # Posterior Predictive For Second Group
   mu.cut <- cut(mu)
   lambda.cut <- cut(lambda)
   predphi ~ dnorm(mu.cut,lambda.cut)I(0,1)
   # Data Follow Binomial With Rate Given By Each Person's Group Assignment
   for (i in 1:p){
      k[i] ~ dbin(theta[i,z1[i]],n)
   }
}
```

The code `ExamsQuizzes_2.m` (Matlab) or `ExamsQuizzes_2.R` (R) makes inferences about group membership, the success rate of each person the knowledge group, and the mean and standard deviation of the over-arching Gaussian for the knowledge group.

Exercises

1. Compare the results of the hierarchical model with the original model that did not allow for individual differences.
2. Interpret the posterior predictive distribution given by the variable `predphi`. How does this distribution relate to the posterior distribution for `mu`?
3. What does the posterior distribution for the variable `theta[1,2]` mean?
4. In what sense could the latent assignment of people to groups in this case study be considered a form of model selection?
5. What is the problem with assuming a Gaussian group distribution on a rate parameter? Can you think of a solution to this problem?

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.

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 the question will be answered correctly. A very simple model involves assuming 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.

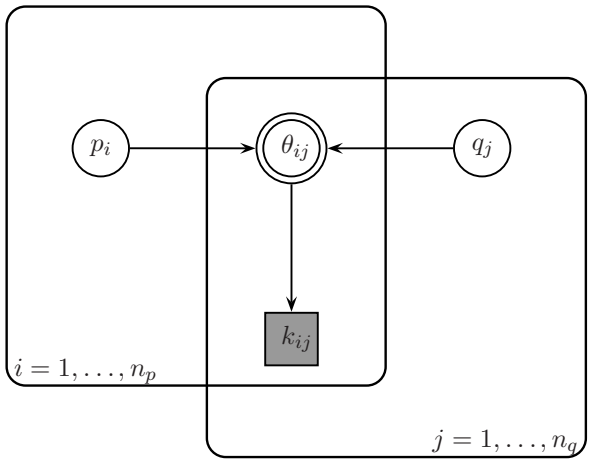
A graphical model that implements this idea is shown in Figure 6.3. Under the model, if the ith person's probability of listening is p_i, and the jth question's probability of being answered correctly if the relevant information is heard is q_j, then the probability the ith person will answer the jth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

```
# Twenty Questions
model {
```

Table 6.1: Correct answers (ones) and incorrect answers (zeros), 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



$p_i \sim \text{Beta}(1, 1)$

$q_j \sim \text{Beta}(1, 1)$

$\theta_{ij} = p_i q_j$

$k_{ij} \sim \text{Bernoulli}(\theta_{ij})$

Figure 6.3: Graphical modeling for inferring the rate people listened to a lecture, and the difficulty of the questions.

```
# 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 `ExamsQuizzes_3.m` (Matlab) or `ExamsQuizzes_3.R` (R) makes inferences about the data in Table 6.1 using the model.

Exercises

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?

2. Once that is done, we get to the fun bit. We now suppose that three of the answers were not recorded. Think of a Scantron[1] with coffee spilled on it being eaten by a dog. Our new data set, with missing data, now take the form shown in Table 6.2.

Table 6.2: Correct answers (ones) and incorrect answers (zeros), for 10 people on 20 questions, with three missing entries shown by question marks.

	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

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/R code to see how all of this is implemented in the second dataset. Run the script, and interpret the posterior distributions for the three missing values. Are they reasonable inferences? Finally, think of a more realistic application for inferring missing values in cognitive modeling than dogs eating coffee flavored Scantrons.

[1] A machine-readable form on which students mark answers to academic test questions.

6.4 The Two Country Quiz

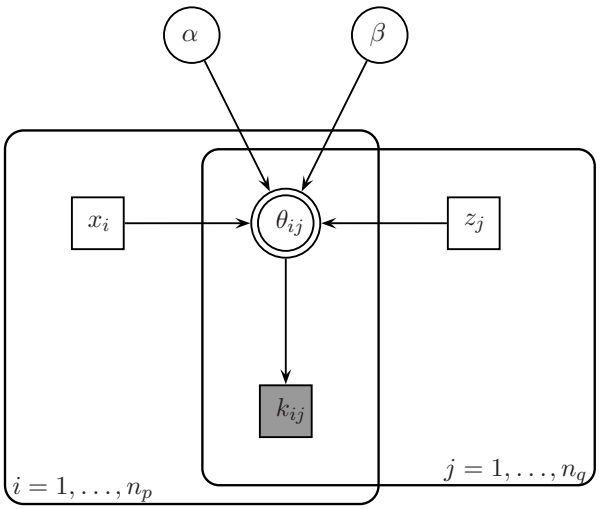
Suppose a group of people take a historical quiz, and each answer for each person is scored as correct or incorrect. Some of the people are Thai, and some are Moldovan. Some of the questions are about Thai history, and would be very likely to be known by any Thai person, but very unlikely to be known by people from outside the region. The rest of the questions are about Moldovan history, and would be very likely to be known by any Moldovan, but not by others.

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 answers (ones) and incorrect answers (zeros), for eight people on eight questions in the country quiz.

	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 will be correct with high probability. For those where a person is being asked about the other country will have a very low probability of being correct.



$$\alpha \sim \text{Beta}(1, 1)$$
$$\beta \sim \text{Beta}(1, 1)$$
$$x_i \sim \text{Bernoulli}(1/2)$$
$$z_j \sim \text{Bernoulli}(1/2)$$
$$\theta_{ij} = \begin{cases} \alpha & \text{if } x_i = z_j \\ \beta & \text{if } x_i \neq z_j \end{cases}$$
$$k_{ij} \sim \text{Bernoulli}(\theta_{ij})$$

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

A graphical model that implements this idea is shown in Figure 6.4. The rate α is the (high) probability

of a person from a country correctly answering a question about their country's history. The rate β is the (low) probability of a person correctly answering a question about the other country's history. 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 follows a Bernoulli distribution with rate θ_{ij}.

The following code implements the graphical model in WinBUGS.

```
# The Two Country Quiz
model {
   # Probability Of Not Forgetting And Guessing
   alpha ~ dbeta(1,1) # Not Forgetting
   beta ~ dbeta(1,1)  # Guessing
   # Group Membership For People and Questions
   for (i in 1:np){
      pz[i] ~ dbern(0.5)
      pz1[i] <- pz[i]+1
   }
   for (j in 1:nq){
      qz[j] ~ dbern(0.5)
      qz1[j] <- qz[j]+1
   }
   # Probability Correct For Each Person-Question Comination By Groups
   # High If Person Group Matches Question Group
   # Low If No Match
   for (i in 1:np){
      for (j in 1:nq){
         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:np){
      for (j in 1:nq){
         k[i,j] ~ dbern(theta[i,j,pz1[i],qz1[j]])
      }
   }
}
```

The code `ExamsQuizzes_4.m` (Matlab) or `ExamsQuizzes_4.R` makes inferences about the data in Table 6.3 using the model.

Exercises

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

2. It is actually quite possible the result you get from this analysis changes from sampling run to sampling run, and some of them may be counter-intuitive. The basic problem is a common one for these sorts of mixture models of *model indeterminacy*. The probability α is used whenever $x_i = z_j$. If this corresponds to a Thai person answering a Thai question, then α should be high, as we expect. But there is nothing stopping the model from coding Thai people as $x_i = 1$ and Moldovan questions as $z_j = 1$, in which case α will be low. Effectively, with this coding, α and β will swap roles. Overall,

there are four possibilities (two ways people can be encoded, by two ways questions can be encoded). Our semantics of α being knowledge-based and β being ignorance-based will apply for 2 of these 4 possible encodings, but will be reversed for the other two. The core problem is that, while we give semantics to α and β in our description of the model, they are statistically defined the same way. This is the indeterminacy. How can this problem can be overcome, not just by being flexible in interpretation, but by defining the statistical model itself more carefully?

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 answers (ones) and incorrect answers (zeros), for eight people, and three late people, on eight questions in the country quiz.

	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	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

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 answers (ones) and incorrect answers (zeros), for eight people, three late people, and ten new people, on eight questions in the country quiz.

	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	?	?	?	?	?	?	?	?

CHAPTER 7

CONVERGENCE

Throughout the previous chapters, we have tacitly assumed that our single MCMC chain was drawing samples from the posterior distribution at the very outset. This does not always happen, and it is important that you are aware of the different methods that can be used to verify that the MCMC samples you base your inference on are indeed samples that come from the posterior distribution.

Indeed, you may have already wondered about the fact that a few chains appeared to show some initial dependence on their starting values, as illustrated in Figure 7.1.

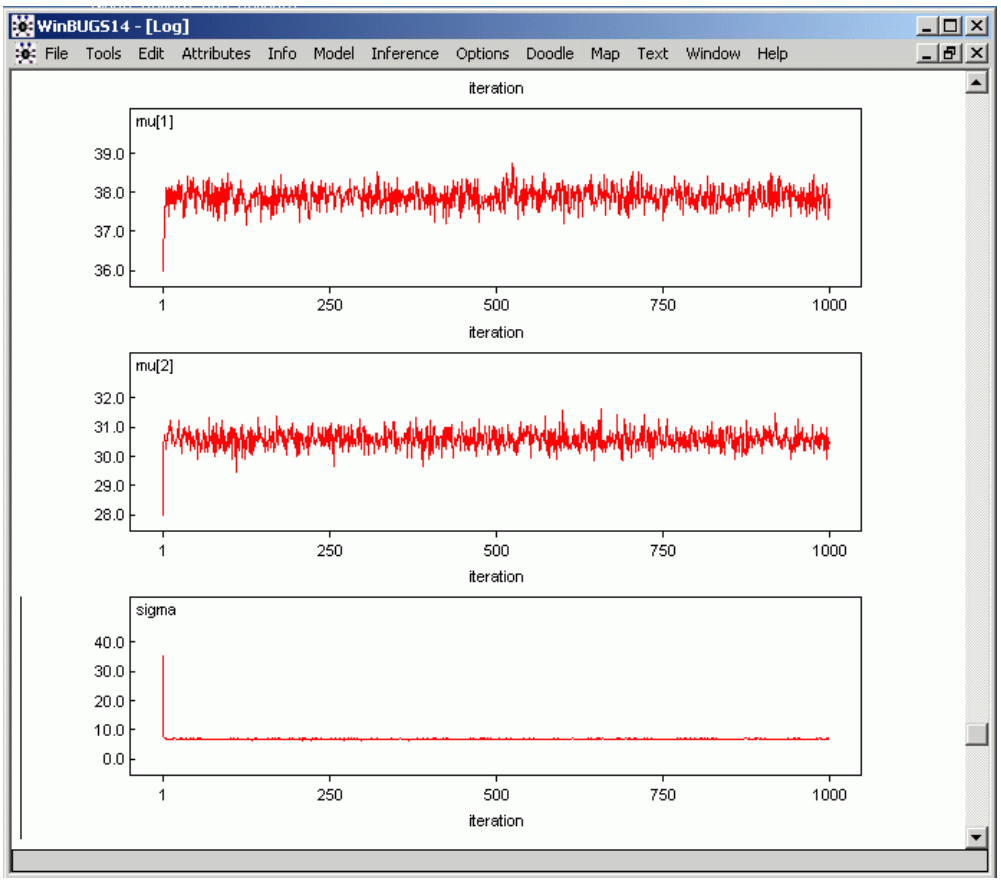


Figure 7.1: Brief dependence on initial values for parameters from the "Change Detection in Time Series Data" example discussed earlier.

In this example, the initial lack of convergence is very short-lived, and consequently the inference is hardly affected at all. The next example demonstrates why lack of convergence can be a real concern.

7.1 A Hierarchical Weighted Average Model and the Conjunction Fallacy

When asked to combine probability statements, people sometimes behave irrationally. Consider for instance the famous example from Tversky and Kahneman (1983):

> *Linda is 31 years old, single, outspoken, and very bright. She majored in philosophy. As a student, she was deeply concerned with issues of discrimination and social justice and also participated in anti-nuclear demonstrations. What is the probability of each of the following?*
>
> 1. Linda is a bank teller. (A)
> 2. Linda is active in the feminist movement. (B)
> 3. Linda is a bank teller and is active in the feminist movement. (A&B)

Suppose that you assign to Statement 1 a probability of .05, that is, $P^A = .05$, and you assign to Statement 2 a probability of .5, that is, $P^B = .5$. If events A and B are independent, a rational agent would now determine the probability of the conjunction, Statement 3, to be the product of the two constituent probabilities: $P^{AB} = P^A \times P^B = .05 \times .5 = .025$. Regardless of whether the events are completely independent (in the Linda example, they are probably not), the key insight is that—from a rational perspective—the probability of the conjunction has to be *less* than the probability of the least likely constituent.

Confronted with the Linda problem, however, many people make an irrational judgment and can state that, for instance, $P^A = .05$, $P^B = .5$, and $P^{AB} = .1$. This irrational judgment is known as the *conjunction error*, and several models and theories have been proposed to explain it.

Here we focus on a particularly simple model, which states that people determine the conjunctive probability, P^{AB} by taking a weighted average (Carlson & Yates, 1989; Fantino, Kulik, Stolarz–Fantino, & Wright, 1997; Nilsson, Winman, Juslin, & Hansson, 2009):

$$P_i^{AB} = \beta_i P_i^A + (1 - \beta_i) P_i^B, \tag{7.1}$$

where the subscript i indicates that the associated quantities may differ from one individual to the next, P_i^B is higher than P_i^A, and $\beta \in [0, 1]$.

In a recent study, Nilsson et al. (2009) reported and analyzed data from an experiment that featured 33 participants, each of which completed 90 Linda-like problems in which it was required to assess the probability of two constituents as well as the probability of the conjunction.

To analyze these data in WinBUGS, we start by building a graphical model, shown in Figure 7.3. In the model, each participant has its own rate parameter β_i. We would like to assume that these individual parameters are drawn from some group-level Normal distribution (as in the earlier example "Exam Scores with Individual Differences"), but there is a complication: the rate scale only ranges from 0 to 1, whereas the Normal distribution ranges from $-\infty$ to $+\infty$.

One solution to this problem is to first transform the rate scale to a different scale. Figure 7.2 shows one such transformation, the so-called *probit* transformation. The probit transform is the inverse cumulative distribution function of the standard Normal distribution, so that, for instance, a rate of $\beta_i = 0.5$ maps onto a probit rate of $\phi_i = 0$, and a rate of $\beta_i = 0.975$ maps onto a probit rate of $\phi_i = 1.96$. The main point is that, in contrast to the rate scale, the probit scale covers the entire real line. Another useful fact is that a standard Normal distribution on the probit scale, $\phi \sim N(0, 1)$ corresponds to a flat, uniform distribution on the rate scale, $\beta \sim U(0, 1)$.

Returning to the graphical model, we assume that each participant has its own rate parameter β_i, and we assume that the probit-transformed parameters ϕ_i come from a Normal group distribution with mean μ and standard deviation σ. From the estimated β_i and the observed P_{ij}^A and P_{ij}^B follows a deterministic prediction about the observed conjunction P_{ij}^{AB}, given by Equation 7.1. In WinBUGS, variables can only fulfill a single function, and they cannot both be observed and deterministically determined (try it and you will receive the error message "multiple definitions of node $PAB[1, 1]$"). To get around this limitation, we state that P_{ij}^{AB} is Normally distributed with a mean equal to Equation 7.1, and a very high precision, thereby mimicking the deterministic prediction from the model.

The following code implements the graphical model in WinBUGS.

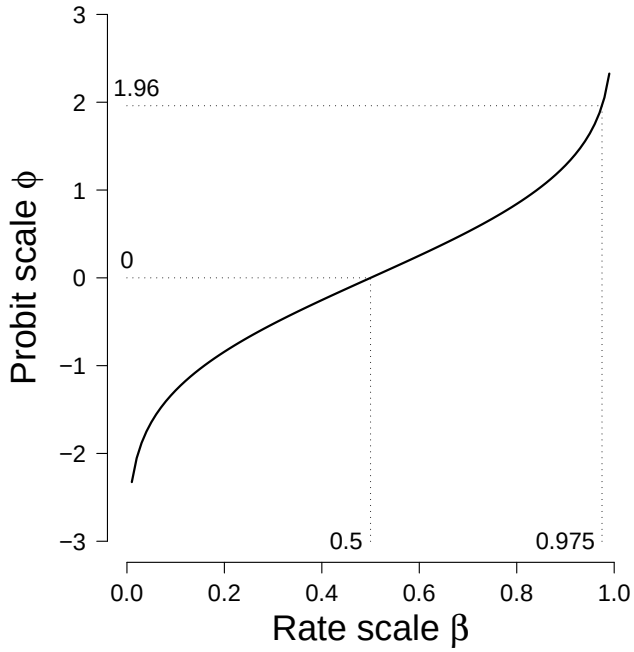


Figure 7.2: The probit transformation. $\beta = \Phi(\phi)$ and $\phi = \Phi^{-1}(\beta)$, where Φ denotes the cumulative distribution function of the standard Normal distribution.

```
model
{

for (i in 1:nsubj)
{
# Each Individual i Has a Personal Beta
# We First Transform the Beta[i]'s to the Probit Scale,
# and Call These betaphi[i]'s:
   beta[i] <- phi(betaphi[i])

# Personal betaphi's Come From a Group-Level Normal:
   betaphi[i] ~ dnorm(muphi,tauphi)
}
# Priors for the Group-Level Normal:
muphi   ~ dnorm(0,1)
tauphi <- pow(sigmaphi,-2)
sigmaphi ~ dunif(0,10)

# Obtain the Mean on the Probability Scale
mu <- phi(muphi)

# Derive WA Predictions:
for (i in 1:nsubj)
{
   for (j in 1:nitem)
   {
      theta[i,j] <- ((beta[i]*PA[i,j])+((1-beta[i])*PB[i,j]))
      PAB[i,j] ~ dnorm(theta[i,j], lambda)
```

```
   }
}
# The First Line Above Computes the Model Predictions theta[i,j];
# The Second Line Connects theta[i,j] to the Data
# Because theta[i,j] is Really a Point Prediction, We'll Use
# "dnorm(theta[i,j],lambda)", Where the Precision lambda is
# Set Very High:
lambda <- 1000
}
```

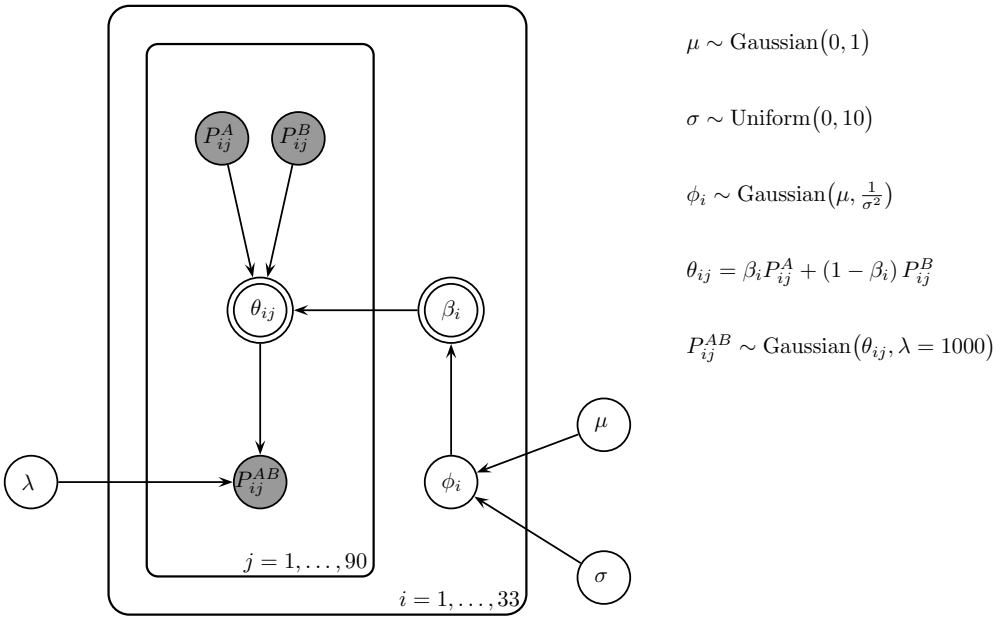


Figure 7.3: Graphical model for the hierarchical weighted average model that explains the conjunction fallacy.

Exercises

1. When you consider the graphical model shown in Figure 7.3, is there a particular source of variability that is potentially important but left unaccounted for?
2. What strong prediction does the weighted average model make with respect to conjunctions P^{AB}?

7.2 Assessing and Improving Convergence

The code `Convergence_1.m` (Matlab) or `Convergence_1.R` (R) makes inferences about the data. Run it. To study convergence, we have implemented three chains, and gave them different starting values. The result should look similar to those shown in Figure 7.4. It is quite clear that the chains behave very differently from what we have seen before. Look at this figure for a while, and write down the different features that all suggest that the sampling process has not yet converged. What measures can you think of to remedy the situation?

To check whether or not the sampling process has converged to the posterior distribution, we can use various tests, including the following (see also Gelman, 1996; Gelman & Hill, 2007):

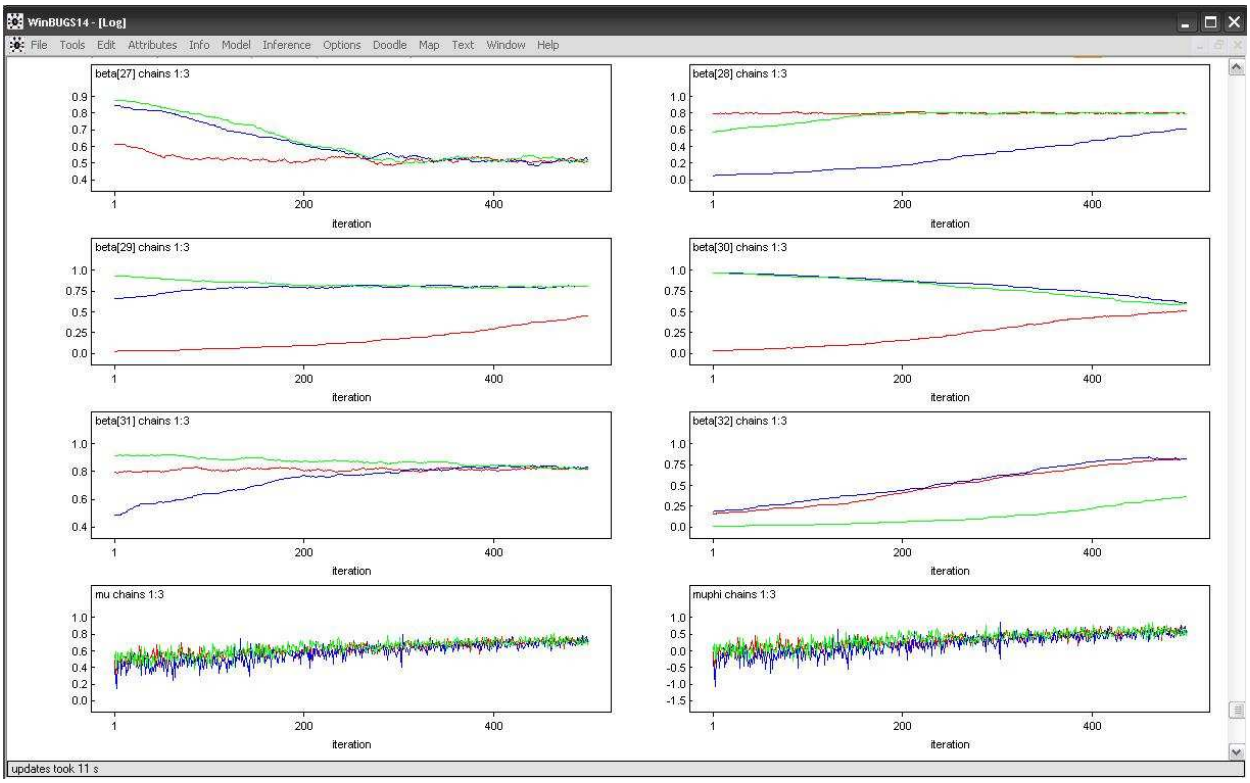


Figure 7.4: Long-term dependence on initial values for parameters from the weighted average model.

1. When the sampling process has converged, chains with substantially different starting values should be indistinguishable from each other (this is why it is important to run more than a single chain, especially for non-standard models).

2. When the sampling process has converged, chains should vary around a constant mean (i.e., there should not be a slow drift up or down).

3. When the sampling process has converged, each individual chain should look like a "fat hairy caterpillar". This occurs when successive values are relatively independent, or when a chain consists of relatively many samples.

4. Formal tests for convergence also exists. One of the most prominent of those is the Gelman and Rubin (1992) $\hat{R}$ statistic that compares the between-chain variance to the within-chain variance. When the chain has converged, $\hat{R}$ should be very close to 1. As a rule of thumb, values higher than 1.1 are (deeply) suspect.

   When you use the R2WinBUGS program, $\hat{R}$ is automatically available after you execute the `samples = bugs(...)` command—just type `plot(samples)` and you should see output similar to that shown in Figure 7.5. It is noteworthy that the $\hat{R}$ for the `muphi` parameter is close to 1.06 (in R2WinBUGS, this can be seen by typing `samples$summary`); this suggests possible convergence, but Figure 7.4 suggests that convergence is obtained only for the final 100 samples or so.

Lack of convergence can originate from different sources: the data (too little, too much in violation of the assumed model), the model (misspecified, too many parameters, too much dependence between the parameters), or the prior (unrealistic). Consequently, lack of convergence may be addressed in several ways. The most elegant way is by slightly changing the model formulation through reparameterizations that speed

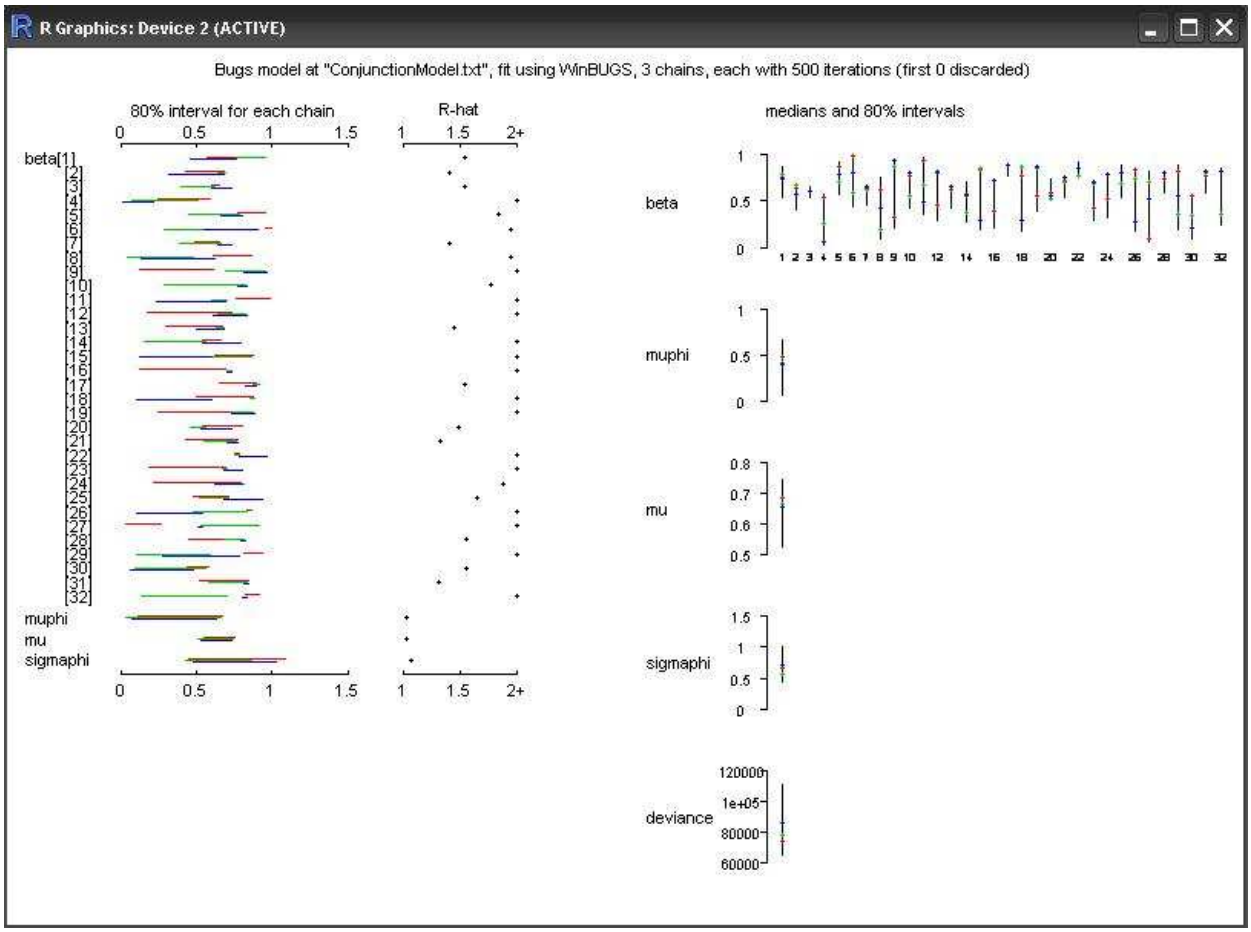


Figure 7.5: $\hat{R}$ convergence measures for the samples from the weighted average model shown in Figure 7.4, as obtained from the R2WinBUGS package.

up convergence (e.g., Gelman & Hill, 2007). You can also check the data for anomalies, and perhaps use transformations to achieve a better match between the data and the model.

But before you start jumping through hoops to achieve better convergence, we advise you to try two easy methods first: (1) eliminate the first m iterations of each chain as "burn-in"— usually $m = 1000$ does the trick; (2) take more samples. This method is not particularly sophisticated, but when your chain is moving relatively slowly through the posterior distribution, you simply have to wait longer to obtain a set of representative samples.

Exercises

1. Change the code in `Convergence_1.m` (Matlab) or in `Convergence_1.R` (R) to draw 10,000 samples with 1000 samples burn-in. Go get yourself a cup of coffee and await the results. Have the chains converged? Did the posterior of `mu` change compared to what you had before?
2. What do you notice with respect to the range of the individual beta's? What does this mean for the conjunction?
3. When you use the R2WinBUGS package, you also have access to the Coda package (Plummer, Best, Cowles, & Vines, 2006), which makes it easy to assess convergence, both formally and informally. The relevant code is available as `Convergence_2.R`. Study and run this code.

4. When you use the R2WinBUGS package, type `help(package="coda")` at the R prompt. Take a look at some of the other formal tests of convergence, and give them a try.
5. Can you suggest some ways to improve the model? (this is a more general question that is not specifically related to convergence)
6. How would you assess whether the model provides a satisfactory account of the data? (again, this is a more general question that is not specifically related to convergence)

Part II

Parameter Estimation for Cognitive Models

CHAPTER 8

INDIVIDUAL DIFFERENCES IN MEMORY RETENTION

Finding a lawful relationship between memory retention and time is about the oldest cognitive modeling question, going back to Ebbinghaus in the 1880s. The usual experiment involves given people (or animals) many items of information on a list, and then testing their ability to remember items from the list after different periods of time have elapsed. Various mathematical functions, usually with psychological interpretations, have been proposed as describing the relationship between time and the level of retention. These include models like exponential decay, power, and hyperbolic functions (Rubin & Wenzel, 1996; Rubin, Hinton, & Wenzel, 1999).

Our example relies on a simplified version of the exponential decay model. The model assumes that the probability an item will be remembered after a period of time t has elapsed is $\theta_t = \exp(-\alpha t) + \beta$, with the restriction $0 < \theta_t < 1$. The α parameter corresponds to the rate of decay of information. The β parameter corresponds to a baseline level of remembering that is assumed to remain even after very long time periods. This model may or may not be regarded as a serious theoretical contender in the memory retention modeling literature, but is useful for simulation and illustrative purposes. Our analyses are based on fictitious data from a potential memory retention study.

Our fictitious data are given in Table 8.1, and relate to 4 subjects tested on 18 items at 10 time intervals: 1, 2, 4, 7, 12, 21, 35, 59, 99, and 200 seconds. The number of items tested and the first 9 time intervals are those used by Rubin et al. (1999), in an attempt to consider data that realistically could be measured in a psychological experiment. Each datum in Table 8.1 simply counts the number of correct memory recalls for each subject at each time interval. Included in Table 8.1 are missing data, shown by '–' symbols, so that we can test the prediction and generalization properties of models. All of the subjects have missing data for the final time period of 200 seconds, so we can test the ability of the model to generalize to new measurements. For Subject 4, there are no data at all, so we can test the ability of models to generalize to new subjects.

8.1 No Individual Differences

The graphical model for our first attempt to account for the data is shown in Figure 8.1. In the graphical model, nodes represent variables of interest, and the graph structure is used to indicate dependencies between the variables, with children depending on their parents. We use the conventions of representing continuous variables with circular nodes and discrete variables with square nodes, and unobserved variables without shading and observed variables with shading. For unobserved variables, we distinguish between stochastic variables with single borders and deterministic variables with double borders. We also use plate notation, enclosing with square boundaries subsets of the graph that have independent replications in the model.

The model in Figure 8.1 assumes is that every subject has the same retention curve, and so there is one true value for the α and β parameters. The outer plate with $j = 1, \ldots, T$ corresponds to the $T = 10$ different time periods, whose values are given by the observed t_j variable. Together with the α and β parameters, these time periods define the probability and item with be remembered. The probability of remembering for the jth time period is the deterministic θ_j node.

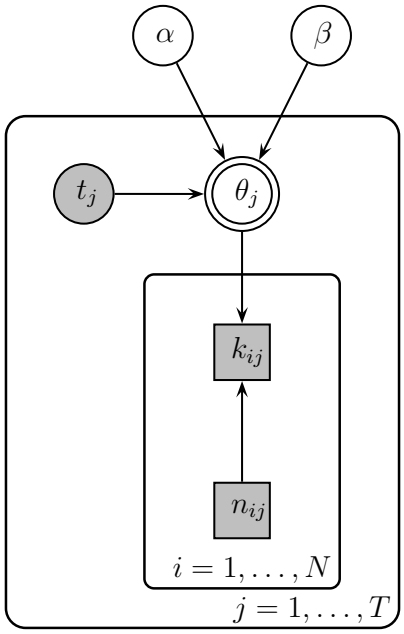
The inner plate with $i = 1, \ldots, N$ corresponds to the $N = 4$ subjects. Each has the same probability of recall at any given time period, but their experimental data, given by the success counts k_{ij} and (potentially) the number of trials n_{ij} vary, and so are inside the plate. For the data in Table 8.1, the k_{ij} data are the counts of remembered items, and $n_{ij} = 18$ because 18 items were presented for every subject at every time interval. The success counts are Binomially distributed according to the success rate and number of trials.

The following code implements the graphical model in WinBUGS.

```
# Retention With No Individual Differences
model {
   # Data Follow A Binomial Distribution
```

Table 8.1: Fictitious memory retention data, giving the number out of 18 items correctly recalled for 3 subjects over 9 time intervals, and including an extra retention interval of 200 secs and an extra subject as missing data.

	Time Interval (secs)									
Subject	1	2	4	7	12	21	35	59	99	200
1	18	18	16	13	9	6	4	4	4	–
2	17	13	9	6	4	4	4	4	4	–
3	14	10	6	4	4	4	4	4	4	–
4	–	–	–	–	–	–	–	–	–	–



$$\alpha \sim \text{Uniform}(0, 1)$$

$$\beta \sim \text{Uniform}(0, 1)$$

$$\theta_j = \exp(-\alpha t_j) + \beta \quad 0 < \theta_j < 1$$

$$k_{ij} \sim \text{Binomial}(\theta_j, n_{ij})$$

Figure 8.1: Graphical model for the exponential decay model of memory retention, assuming no individual differences.

```
for (i in 1:ns){
   for (j in 1:nt){
      k[i,j] ~ dbin(theta[i,j],n[i,j])
   }
}
# Retention Rate At Each Lag For Each Subject Decays Exponentially
for (i in 1:ns){
   for (j in 1:nt){
      theta[i,j] <- min(1,exp(-alpha*t[j])+beta)
   }
}
# Priors
alpha ~ dunif(0,1)
beta ~ dunif(0,1)
```

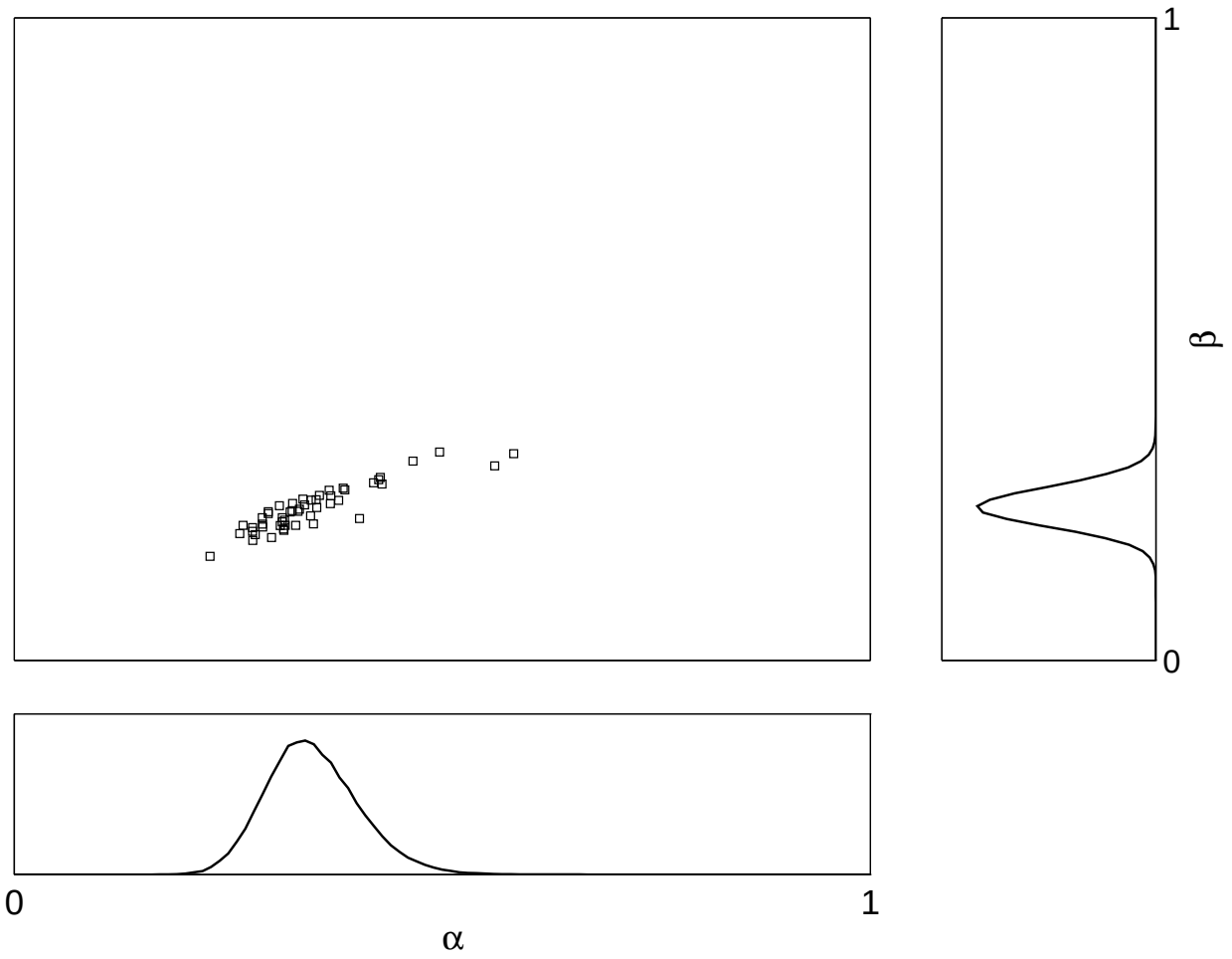


Figure 8.2: The joint and marginal posterior distributions over the decay and permanent retention parameters α and β, for the model that assumes no individual differences.

```
   # Posterior Predictive
   beta.cut <- cut(beta)
   alpha.cut <- cut(alpha)
   for (i in 1:ns){
      for (j in 1:nt){
         predtheta[i,j] <- min(1,exp(-alpha.cut*t[j])+beta.cut)
         predk[i,j] ~ dbin(predtheta[i,j],n[i,j])
      }
   }
}
```

The code `Retention_1.m` (Matlab) or `Retention_1.R` (R) applies the model to the fabricated data, and produces analysis of the posterior and posterior predictive distributions.

The joint posterior distribution over α and β is shown in the main panel of Figure 8.2, as a two-dimensional scatterplot. Each of the 50 points in the scatterplot corresponds to a posterior sample selected at random from the 10^4 available. The marginal distributions of both α and β are shown below and to the right, and are based on all 10^4 samples. The marginals show the distribution of each parameter, conditioned

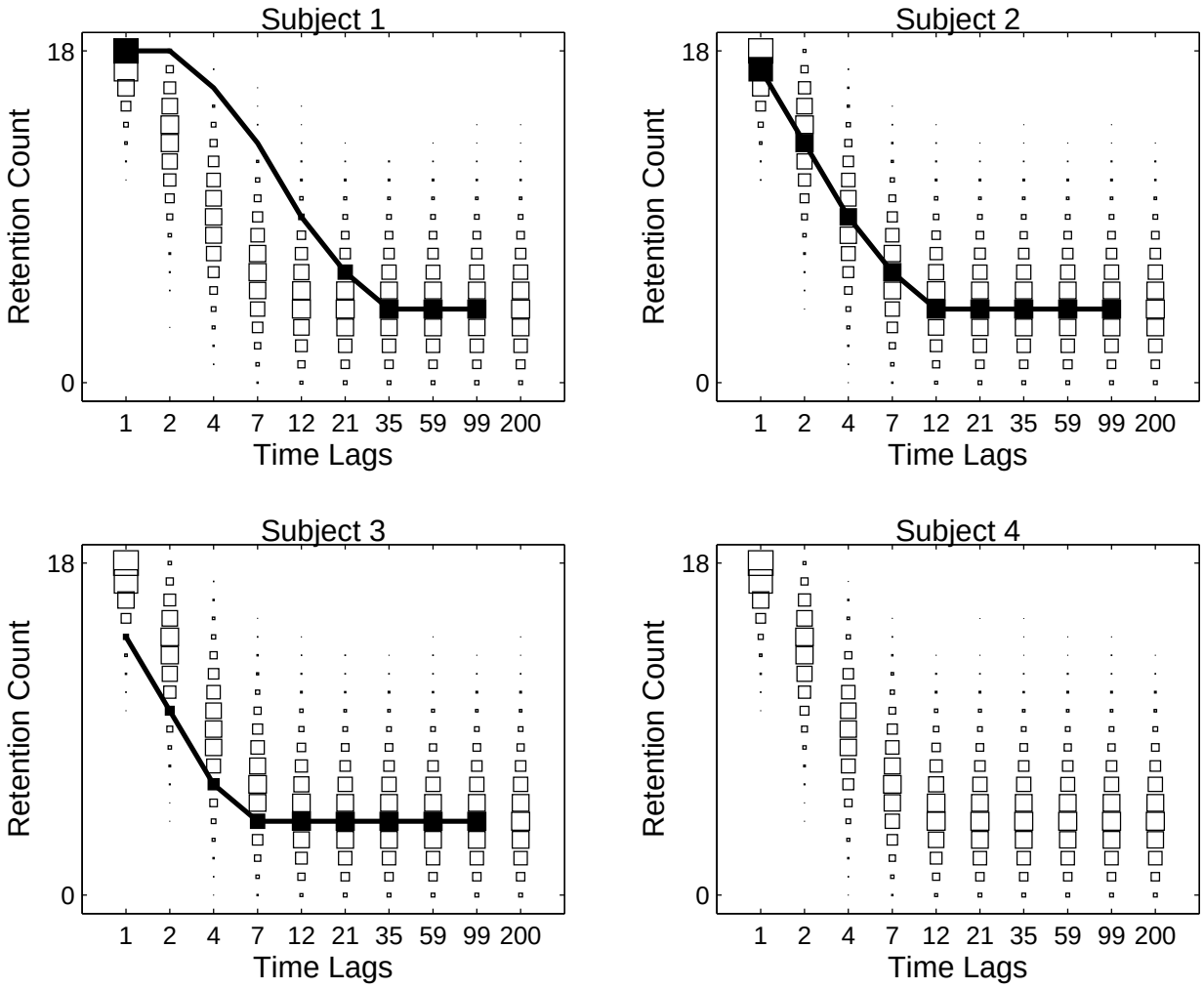


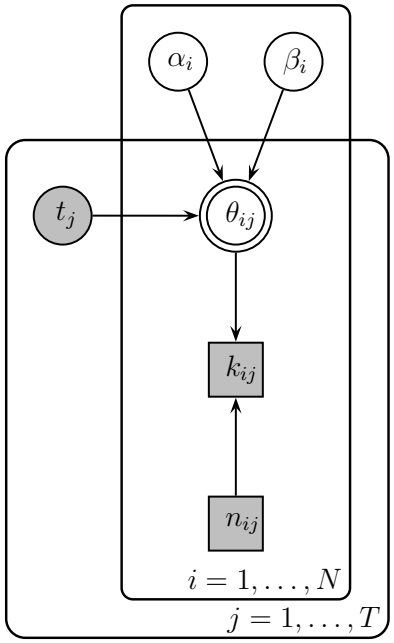
Figure 8.3: The posterior predictive for the model that assumes no individual differences, against the data from the four subjects.

on the data, considered independently (i.e., averaged across) the other parameter.

It is clear from Figure 8.2 that the joint posterior carries more information than the two marginal distributions. If the joint posterior were independent, it would be just the product of the two marginals, and carry no extra information. But the joint posterior shows a mild relationship, with larger values of α generally corresponding to larger values of β. This can be interpreted psychologically as meaning the relatively higher baselines are needed to model the data if relatively greater rates of decay are used.

Figure 8.3 shows the posterior predictive distribution over the number of successful retentions at each time interval. For each subject, at each interval, the squares show the posterior mass given to each possible number of items recalled. These correspond to the models predictions about observed behavior in the retention experiment, based on what the model has learned from the data. Also shown, by the black squares and connecting lines, are the actual observed data for each subject, where available.

The obvious feature of Figure 8.3 is that the current model does not meet a basic requirement of descriptive adequacy. For both Subjects 1 and 3 the model gives little posterior probability to the observed data at many time periods. It predicts a steeper rate of decay than shown by the data of Subject 1, and a shallower rate of decay than shown by the data of Subject 3. After evaluating the model using the posterior



$$\alpha_i \sim \text{Uniform}(0, 1)$$

$$\beta_i \sim \text{Uniform}(0, 1)$$

$$\theta_{ij} = \exp(-\alpha_i t_j) + \beta_i \quad 0 < \theta_j < 1$$

$$k_{ij} \sim \text{Binomial}(\theta_{ij}, n_{ij})$$

Figure 8.4: Graphical model for the exponential decay model of memory retention, assuming full individual differences.

predictive analysis, we can conclude that the assumption of no individual differences is inappropriate. It is important to understand that this conclusion neuters the usefulness of the posterior distribution over parameters, as shown in Figure 8.2. This posterior distribution is conditioned on the assumption that the model is appropriate, and is not relevant when our conclusion is that the model is fundamentally deficient.

Exercises

1. Why is the posterior predictive distribution for all four subjects the same? Are there any (real or fabricated) data that could make the model predict different patters of retention for different subjects? What about if there were massive qualitative differences, such as one subject remembering everything, and the other two remembering nothing?

8.2 Full Individual Differences

A revised graphical model that does accommodate individual differences is shown in Figure 8.4. The change from the previous model is that every subject now has their own α_i and β_i parameters, and that the probability of retention for an item θ_{ij} now changes for both subjects and retention intervals.

The following code implements the graphical model in WinBUGS.

```
# Retention With Full Individual Differences
model {
   # Data Follow A Binomial Distribution
   for (i in 1:ns){
      for (j in 1:nt){
         k[i,j] ~ dbin(theta[i,j],n[i,j])
      }
   }
```

```
   # Retention Rate At Each Lag For Each Subject Decays Exponentially
   for (i in 1:ns){
      for (j in 1:nt){
         theta[i,j] <- min(1,exp(-alpha[i]*t[j])+beta[i])
      }
   }
   # Priors For Each Subject
   for (i in 1:ns){
      alpha[i] ~ dunif(0,1)
      beta[i] ~ dunif(0,1)
   }
   # Posterior Predictive
   for (i in 1:ns){
      alpha.cut[i] <- cut(alpha[i])
      beta.cut[i] <- cut(beta[i])
      for (j in 1:nt){
          predtheta[i,j] <- min(1,exp(-alpha.cut[i]*t[j])+beta.cut[i])
          predk[i,j] ~ dbin(predtheta[i,j],n[i,j])
      }
    }
}
```

The code `Retention_2.m` (Matlab) or `Retention_2.R` (R) applies the model to the fabricated data, and again produces analysis of the posterior and posterior predictive.

The joint posterior distributions for each subject are shown in the main panel of Figure 8.5. Each point the scatterplot corresponds to a posterior sample, with different markers representing different subjects. The first, second, third and fourth subjects use '+', '⋆', 'x' and 'o' markers, respectively. The marginal distributions are shown below and to the right, and use different line styles to represent the subjects.

Figure 8.6 shows the same analysis of the posterior predictive distribution over the number of successful retentions at each time interval, for each subject. It is clear that allowing for individual differences lets the model achieve a basic level of descriptive adequacy for Subjects 1 and 3. The posteriors in Figure 8.5 show that different values for the α decay parameter are used for Subject 1, 2, and 3, corresponding to our intuitions from the earlier analysis.

The weakness in the current model is evident in its predictions for Subject 4. Because each subject is assumed to have decay and permanent retention parameters that are different, the only information the model has about the new subject are the priors for the α and β parameters. The relationships between parameters for subjects that are visually evident in Figure 8.5 are not formally captured by the model. This means, as shown in Figure 8.5, the posteriors for Subject 4 are just the priors, and so the posterior predictive for this subject, shown in Figure 8.6, does not have any useful structure. In this way, the current model fails a basic test of generalizability, since it does not make sensible predictions for the behavior of future subjects.

Intuitively one might want to predict that Subject 4 will be likely to have model parameters represented by some sort of average of Subjects 1 to 3. Carrying this intuition a bit further, one might also want Subjects 1 to 3 to have their highest likelihood parameters closer to their group mean than is the case when choosing individual parameters independently.

Exercises

1. What are the relative strengths and weaknesses of this full individual differences model compared to the earlier no individual differences model? Think about this, because the hierarchical approach we consider next could be argued to combine the best features of both of these approaches.

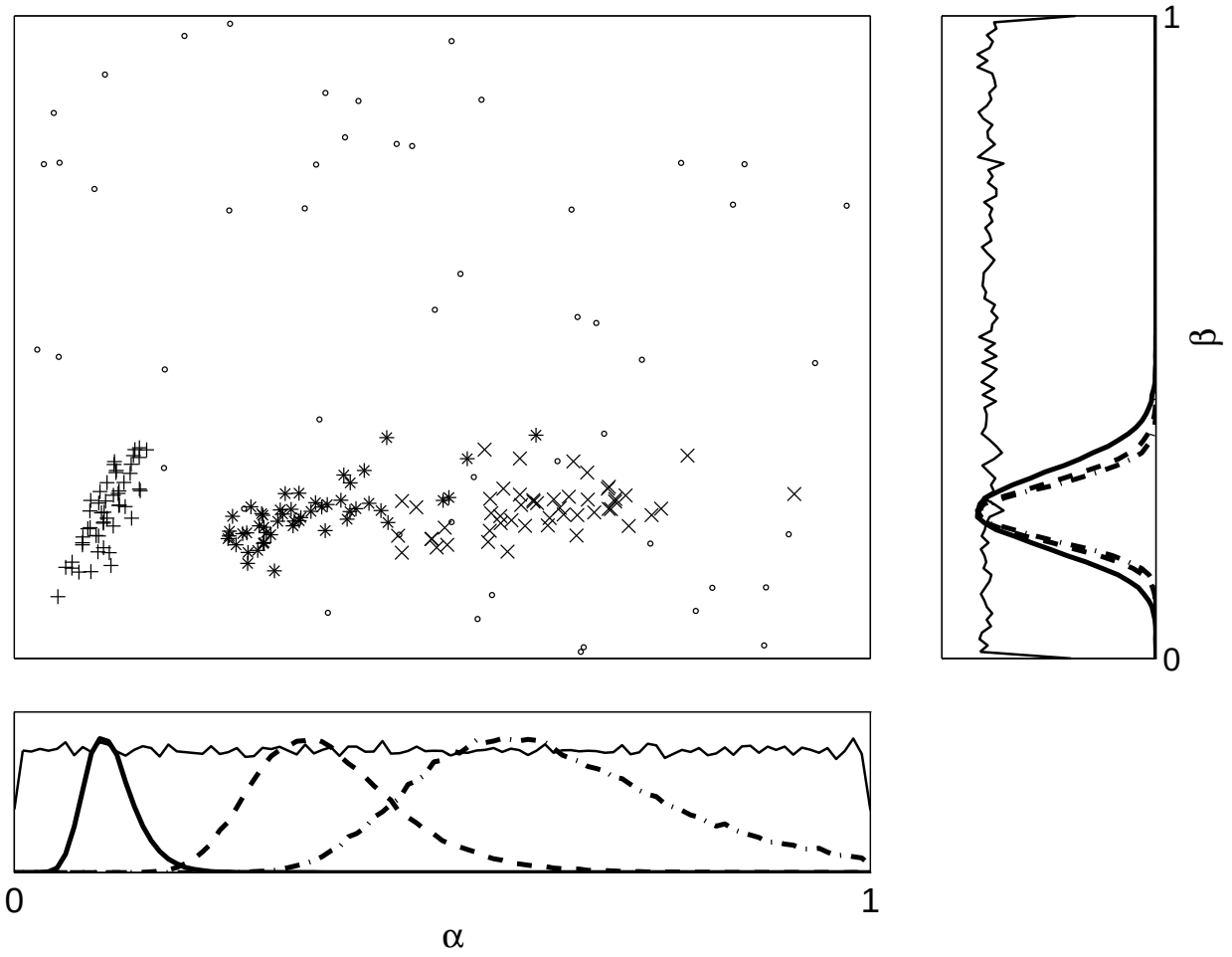


Figure 8.5: The joint and marginal posterior distributions over the decay and permanent retention parameters α and β, for the model that assumes full individual differences.

8.3 Structured Individual Differences

The relationship between the parameters of structures is naturally addressed in a hierarchical model, which is able to represent knowledge at different levels of abstraction in a cognitive model. Just as the data have been assumed to be generated by the latent decay and permanent retention parameters for individual subjects, we now assume that those parameters themselves are generated by more abstract latent parameters that describe group distributions across subjects.

The specific graphical model we used to implement this idea is in Figure 8.7. The key change is that now we are modeling the variation in the different α_i and β_i parameters for each subject, by assuming they have a Gaussian distribution across subjects. This means that the α_i and β_i parameters are now sampled from over-arching Gaussian distributions, themselves with unknown parameters in the form of means μ_α and μ_β and precisions λ_α and λ_β.

Because they are now sampled, the α_i memory decay and β_i permanent retention parameters no longer have priors explicitly specified, but inherit them from the priors on the means and precisions of the Gaussian distributions. It is important to understand that, consequently, inferences made for one subject influence predictions made for another. Since the means and precisions of the group-level distributions are common to all subjects, what is learned about them from one subject affects what is known about another. It is in

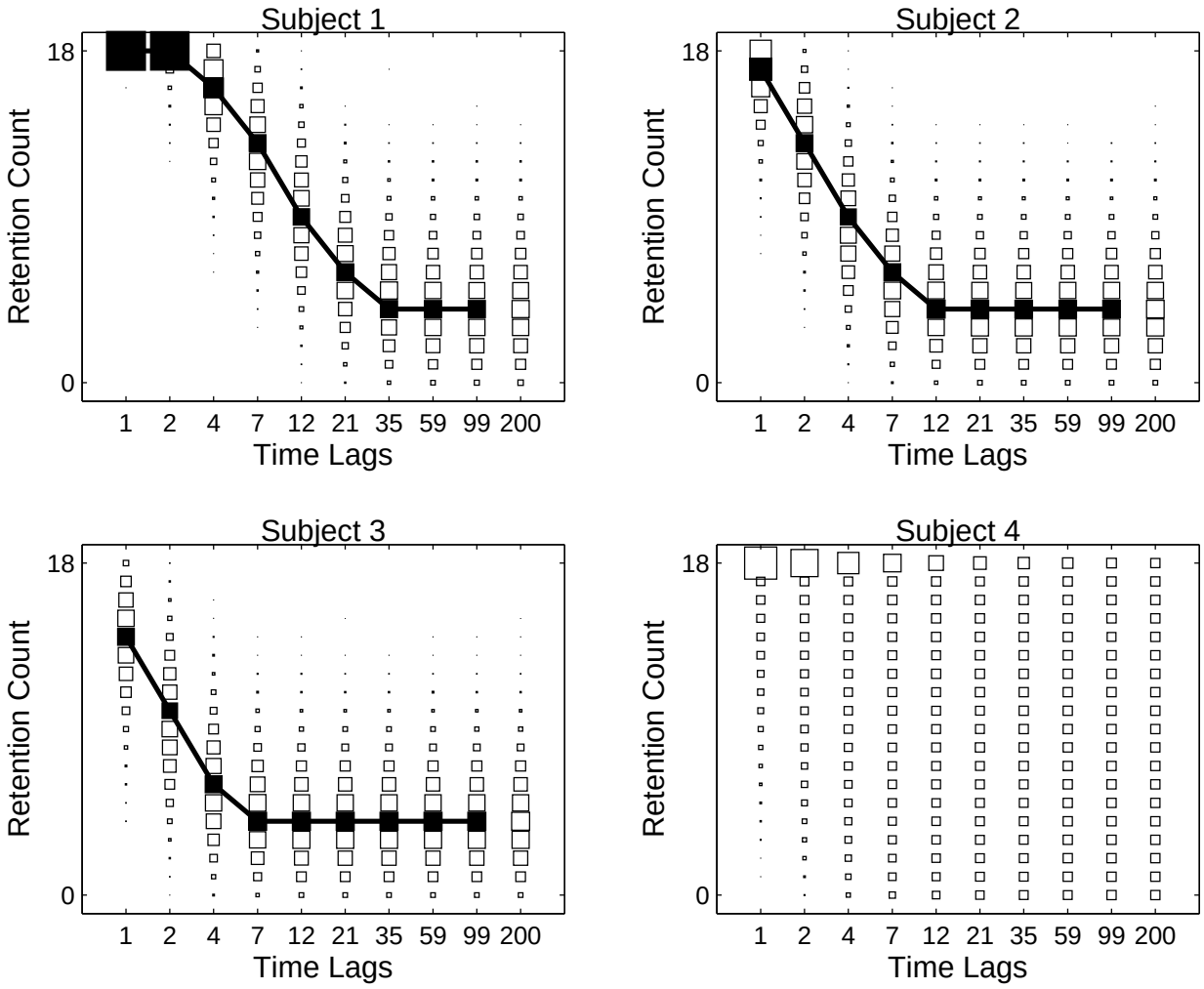
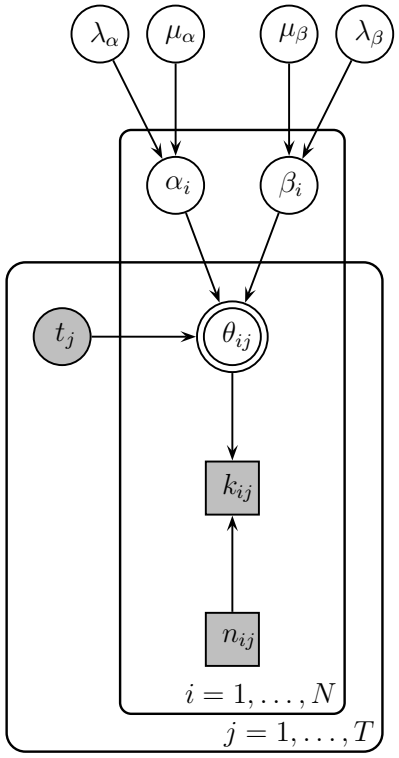


Figure 8.6: Analysis of the posterior predictive for the model that assumes full individual differences, against the data from the four subjects.

this way the hierarchical model formally represents the relationships between subjects.

The following code implements the graphical model in WinBUGS.

```
# Retention With Structured Individual Differences
model{
   # Data Follow A Binomial Distribution
   for (i in 1:ns){
      for (j in 1:nt){
         k[i,j] ~ dbin(theta[i,j],n[i,j])
      }
   }
   # Retention Rate At Each Lag For Each Subject Decays Exponentially
   for (i in 1:ns){
      for (j in 1:nt){
         theta[i,j] <- min(1,exp(-alpha[i]*t[j])+beta[i])
      }
```



$$\mu_\alpha \sim \text{Uniform}(0, 1)$$

$$\lambda_\alpha \sim \text{Gamma}(.001, .001)$$

$$\mu_\beta \sim \text{Uniform}(0, 1)$$

$$\lambda_\beta \sim \text{Gamma}(.001, .001)$$

$$\alpha_i \sim \text{Gaussian}(\mu_\alpha, \lambda_\alpha) \qquad 0 < \alpha_i < 1$$

$$\beta_i \sim \text{Gaussian}(\mu_\beta, \lambda_\beta) \qquad 0 < \beta_i < 1$$

$$\theta_{ij} = \exp\left(-\alpha_i t_j\right) + \beta_i \qquad 0 < \theta_{ij} < 1$$

$$k_{ij} \sim \text{Binomial}(\theta_{ij}, n_{ij})$$

Figure 8.7: Graphical model for the exponential decay model of memory retention, assuming structured individual differences.

```
   }
   # Parameters For Each Subject Drawn From Gaussian Group Distributions
   for (i in 1:ns){
      alpha[i] ~ dnorm(alphamu,alphalambda)I(0,1) # Censor To Valid Range
      beta[i] ~ dnorm(betamu,betalambda)I(0,1) # Censor To Valid Range
   }
   # Priors For Group Distributions
   alphamu ~ dunif(0,1)
   alphalambda ~ dgamma(.001,.001)
   alphasigma <- 1/sqrt(alphalambda)
   betamu ~ dunif(0,1)
   betalambda ~ dgamma(.001,.001)
   betasigma <- 1/sqrt(betalambda)
   # Posterior Predictive For Counts
   for (i in 1:ns){
      alpha.cut[i] <- cut(alpha[i])
      beta.cut[i] <- cut(beta[i])
      for (j in 1:nt){
         predtheta[i,j] <- min(1,exp(-alpha.cut[i]*t[j])+beta.cut[i])
         predk[i,j] ~ dbin(predtheta[i,j],n[i,j])
      }
   }
}
```

The code `Retention_3.m` (Matlab) or `Retention_3.R` (R) applies the model to the fabricated data, and

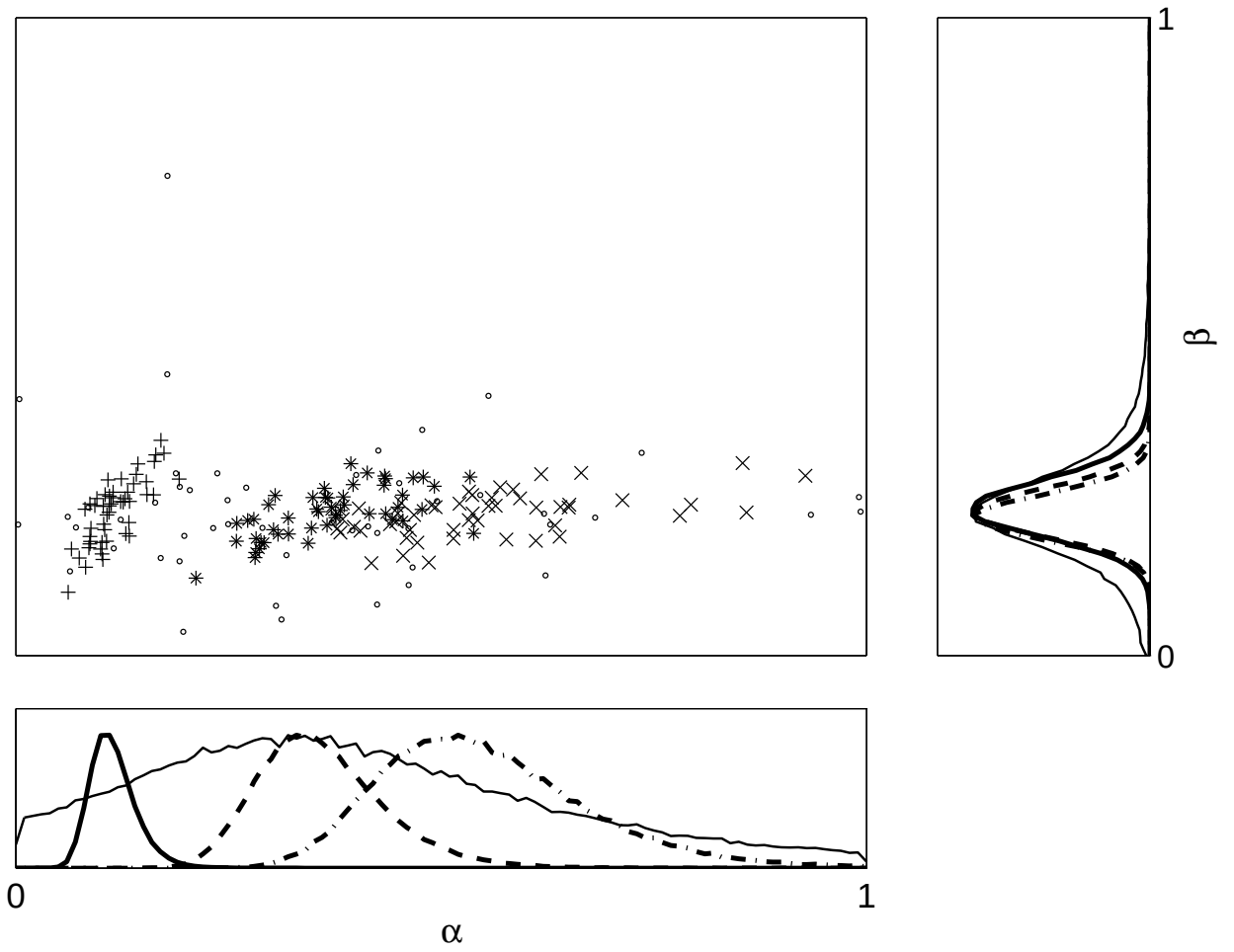


Figure 8.8: The joint and marginal posterior distributions over the decay and permanent retention parameters α and β, for the model that assumes structured individual differences.

again produces analysis of the posterior and posterior predictive.

The joint and marginal posterior distributions for this model are shown in Figure 8.8 are shown using the same markers and lines as before. For Subjects 1, 2, and 3, these distributions are extremely similar to those found using the full individual differences model. The important difference is for Subject 4, who now has sensible posterior distributions for both parameters. For the decay parameter α there is still considerable uncertainty, consistent with the range of values seen for the first three subjects, but for the permanent retention parameter β, Subject 4 now has a much more constrained posterior.

The posterior predictive distributions for each subject under the hierarchical model are shown in Figure 8.9. The predictions remain useful for the first three subjects, and are now also appropriate for Subject 4. This effective prediction for a subject from whom no data have yet been collected arises directly from the nature of the hierarchical model. Based on the data from Subjects 1, 2, and 3, inferences are made about the means and precisions of the group distributions for the two parameters of the retention model. The new Subject 4 has values sampled from the Gaussians with these parameters, producing the sensible distributions in Figure 8.8 that lead to the sensible predictions in Figure 8.9.

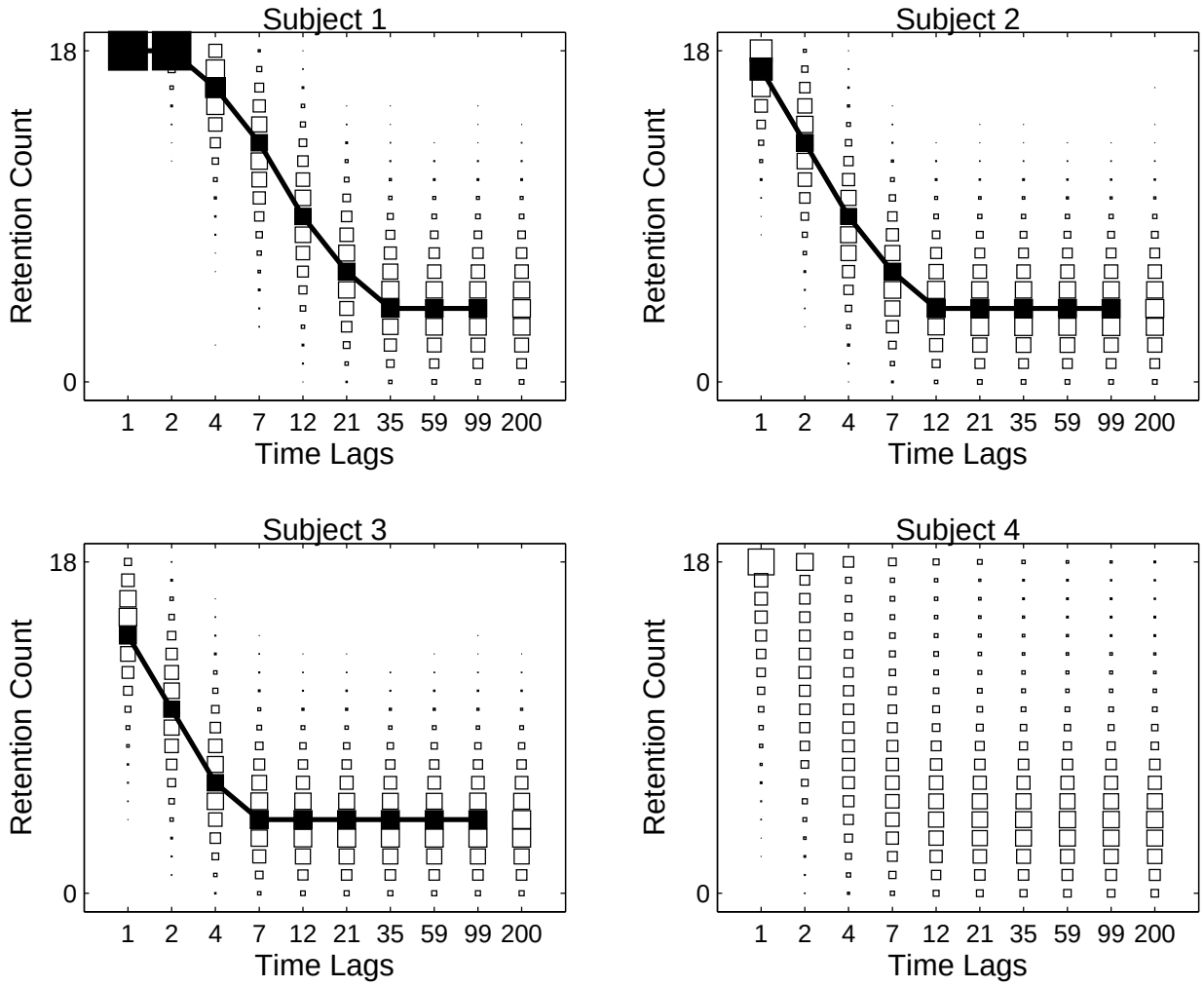


Figure 8.9: Analysis of the posterior predictive for the model that assumes structured individual differences, against the data from the four subjects.

Exercises

1. Think of a psychological model and data, with as little as possible to do with the memory retention example, where this hierarchical approach might be useful.
2. Can you come up with a hierarchical version of the basic model that does not require you to truncate the rate scale using the `I(0,1)` command? Implement this model and see whether your hierarchical model leads to different conclusions than the ones presented here.

CHAPTER 9

SIGNAL DETECTION THEORY

9.1 Standard Signal Detection Theory

Signal Detection Theory (SDT: see D. M. Green & Swets, 1966; MacMillan & Creelman, 2004, for detailed treatments) is a very general, useful, and widely employed method for drawing inferences from data in psychology. It is particularly applicable to two-alternative forced choice experiments, although it can really be applied to any situation that can be conceived as a 2×2 table of counts.

Table 9.1 gives the basic data and terminology for SDT. There are 'signal' trials and 'noise' trials, and 'yes' responses and 'no' responses. When a yes response is given for a signal trial, it is called a 'hit'. When a yes response is given for a noise trial, it is called a 'false alarm'. When a no response is given for a signal trial, it is called a 'miss'. When a no response is given for a noise trial, it is called a 'correct rejection'.

The basic data for a SDT analysis are just the counts of hits, false alarms, misses and correct rejection. If you know the total number of signal and noise trials, which you typically do, then all of the variation in the data is captured by just the hit and false alarm counts.

Table 9.1: Basic Signal Detection Theory data and terminology.

	Signal Trial	Noise Trial
Yes Response	Hit	False Alarm
No Response	Miss	Correct Rejection

The key assumptions of SDT are shown in Figure 9.1, and involve representation and decision-making assumptions. Representationally, the idea is that signal and noise trials can be represented as values along uni-dimensional 'strength' construct. Both types of trials are assumed to produce strengths that vary according to a Gaussian distribution along this dimension. The signal strengths are assumed to be greater, on average, than the noise strengths, and so the signal strength distribution has a greater mean. In the most common equal-variance form of SDT, both the distributions are assumed to have the same variance. The decision-making assumption of SDT is that yes and no responses are produced by comparing the strength of the current trial to a fixed criterion. If the strength exceeds the criterion a yes response is made, otherwise a no response is made.

Figure 9.1 provides a formal version of the equal-variance SDT model. Since the underlying strength scale has arbitrary units, the variances are fixed to one, and the mean of the noise distribution is set to zero. The mean of the signal distribution is d. This makes d a measure of the *discriminability* of the signal trials from the noise trials, because it corresponds to the distance between the two distributions.

The strength value $d/2$ is special, because it is the criterion value at which both signal and noise distributions are equally likely. In this sense, using a criterion of $d/2$ corresponds to unbiased responding. The actual criterion used for responding is denoted k, and distance between this criterion and the unbiased criterion is denoted c. This makes c a measure of *bias*, because it corresponds to how different the actual criterion is from the unbiased one. Positive values of c correspond to a bias towards saying no, and so to an increase in correct rejections at the expense of an increase in misses. Negative values of c correspond to a bias towards saying yes, and so to an increase in hits at the expense of a increase in false alarms.

The SDT model, with its representation and decision-making assumptions, naturally makes predictions about hit rates and false alarm rates, and so maps naturally onto the counts in Table 9.1. In Figure 9.1, the hit rate, h, is shown as the proportion of the signal distribution above the criterion k. Similarly, the false alarm rate, f, is the proportion of the noise distribution above the criterion k.

The usefulness of SDT is that, through this relationship, it is possible to take the sort of data in Table 9.1 and convert the counts of hits and false alarms (which are not independent) into psychologically meaningful

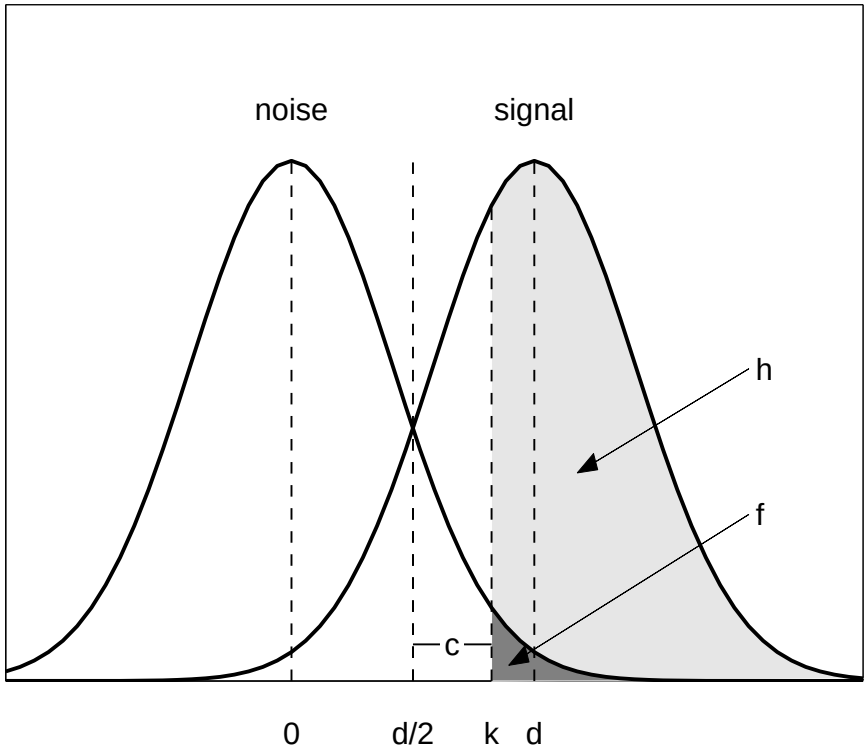


Figure 9.1: Equal-variance Gaussian Signal Detection Theory Framework.

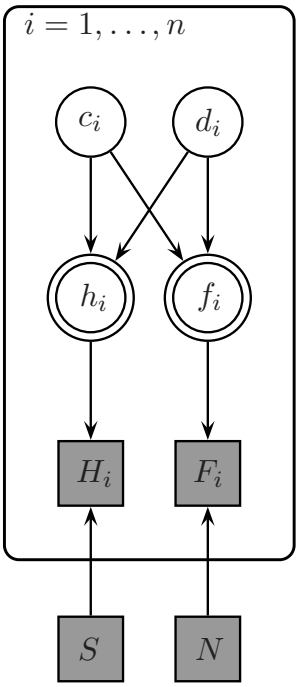
measures of discriminability and bias (which are independent). Discriminability is a measure of how easily signal and noise trials can be distinguished. Bias is a measure of how the decision-making criterion being used relates to the optimal criterion.

A graphical model for inferring discriminability and bias from hit and false alarm counts for $i = 1, \ldots, n$ different cases is shown in Figure 9.2. The hit rates h_i and false alarm rates f_i follow from the geometry of Figure 9.1 as functions of their associated discriminabilities d_i and biases c_i, using the cumulative standard Normal distribution function $\Phi(\cdot)$. The observed counts of hits H_i and false alarms F_i are Binomially distributed according to the hits and false alarm rates, and the number of signal trials S and noise trials N. The priors for discriminability and bias are both Gaussian distributions, carefully constructed[1] to correspond to uniform prior distributions over the hit and false alarm rates.

The following code implements the graphical model in WinBUGS.

```
# Signal Detection Theory
model{
   # Relating observed counts to underlying Hit and False Alarm rates
   for (i in 1:n) {
      # Hit counts per Subject are Binomial
      # Using Hit Rates per Subject and Number of Signal Trials
      HR[i] ~ dbin(h[i],S[i])
      # False Alarm Counts per Subject are Binomial
      # Using False Alarm Rates per Subject and Number of Signal Trials
      FA[i] ~ dbin(f[i],N[i])
      # Number of Signal Trials is Sum of Hit and Miss Counts per Subject
      S[i] <- HR[i]+MI[i]
```

[1] With thanks to Geoff Iverson and some fancy theorem whose name I forget.



$$d_i \sim \text{Gaussian}(0, 1/2)$$
$$c_i \sim \text{Gaussian}(0, 2)$$

$$h_i = \Phi(\tfrac{1}{2}d_i - c_i)$$
$$f_i = \Phi(-\tfrac{1}{2}d_i - c_i)$$

$$H_i \sim \text{Binomial}(h_i, S)$$
$$F_i \sim \text{Binomial}(f_i, N)$$

Figure 9.2: Graphical model for signal detection theory.

```
      # Number of Noise Trials is sum of False Alarm and Correct Rejection Counts per Subject
      N[i] <- FA[i]+CR[i]
   }
   # Reparameterization, Converting Hit and False Alarm Rates
   # to Indices for Discriminability and Bias
   # Assumes Equal-Variance Gaussian Signal Detection Theory Model
   for (i in 1:n) {
      h[i] <- phi(d[i]/2-c[i])
      f[i] <- phi(-d[i]/2-c[i])
   }
   # These Priors over Discriminability and Bias Correspond
   # to Uniform Priors over the Hit and False Alarm Rates:
   for (i in 1:n) {
      d[i] ~ dnorm(0,0.5)
      c[i] ~ dnorm(0,2)
   }
}
```

The code `SDT_1.m` (Matlab) or `SDT_1.R` (R) applies the model to make inferences for three illustrative data sets (described below). Figure 9.3 shows the results produced by the script, plotting posterior distributions for discriminability, bias, hit rate and false-alarm for each data set.

In the first data set, 70 hits and 50 false-alarms are observed in 100 target and 100 distractor trials. Because of the large number of trials, there is relatively little uncertainty surrounding the hit and false-alarm rates, with narrow posteriors centered on 0.7 and 0.5 respectively. Discriminability and bias are also known with some certainty, centered on about 0.5 and -0.25 respectively.

In the second data set, 7 hits and 5 false-alarms are observed in 10 target and 10 distractor trials. These are the same rates of hit and false-alarms of the first situation, but based on many fewer samples. Accordingly, the posterior distributions have (essentially) the same means, but show much greater uncertainty.

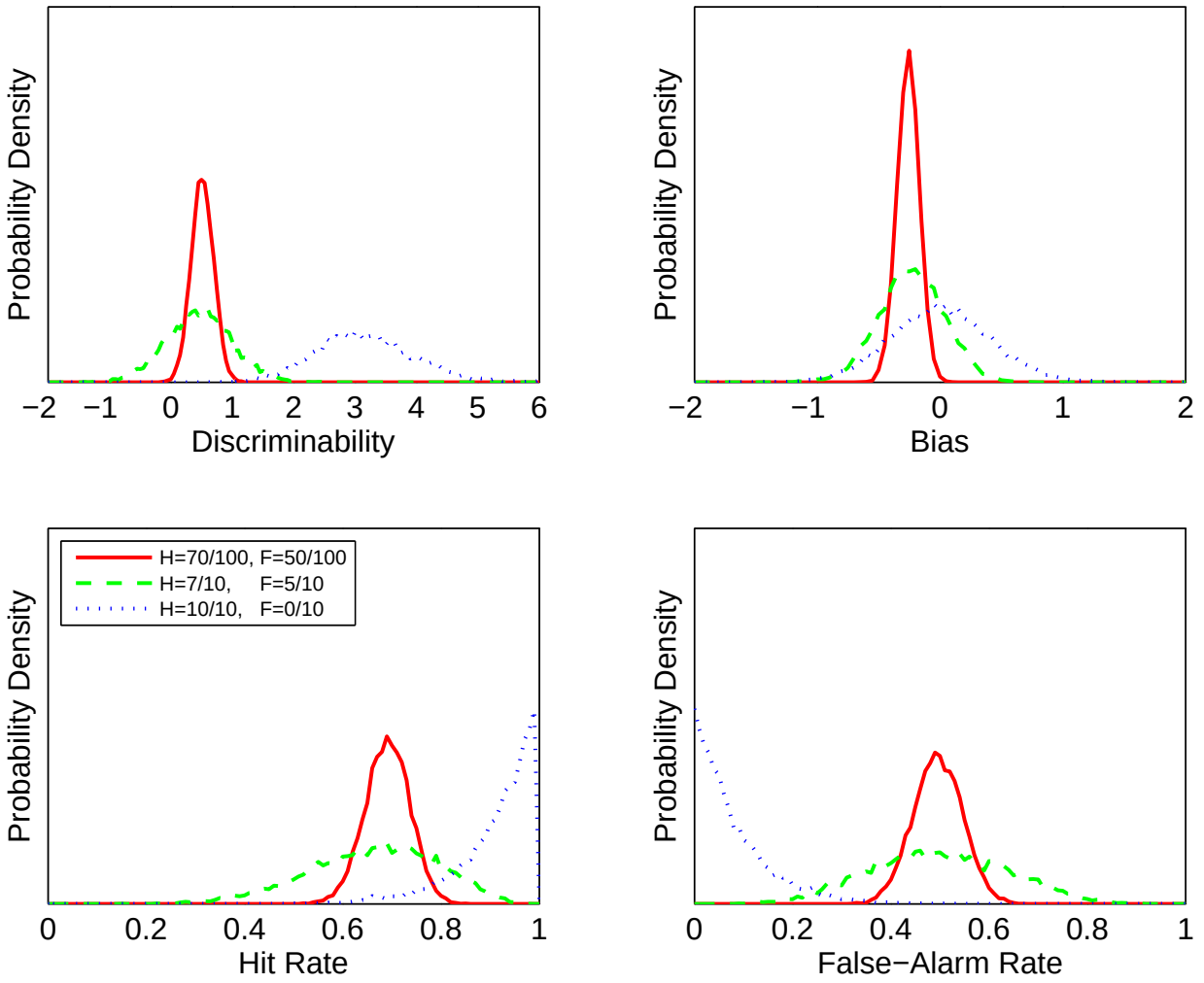


Figure 9.3: Posterior distributions for discriminability, bias, hit rate and false-alarm rate using three illustrative data sets.

In the third data set, perfect performance is observed, with 10 hits and no false-alarms in 10 target and 10 distractor trials. The modal hit and false-alarm rates are 1.0 and 0.0, but other possibilities have some density. Discriminability is certain to be large, although the exact value is not clear. These data provide no information to help estimate bias, and so it retains its prior distribution. These outcome contrasts favorably with traditional frequentist analyses, which have to employ ad-hoc edge corrections to avoid both discriminability and bias being undefined when either no hits or no false alarms are observed.

Exercises

1. Do you feel that the priors on discriminability and bias are plausible, *a priori*? Why or why not? Try out some alternative priors and study the effect that this has on your inference for the data sets discussed above.

2. Lehrner, Kryspin-Exner, and Vetter (1995) report data on the recognition memory for odors of three groups of subjects. Group I had 18 subjects, all with positive HIV antibody tests, and CD-4 counts of 240-700/mm^3. Group II had 19 subjects, all also with positive HIV antibody tests, but with CD-4

counts of 0-170/mm^3. The CD-4 counts is a measure of the strength of the immune system, with a normal range being 500-700/mm^3, so Group II subjects had weaker immune systems. There was also a control group of 18 healthy subjects.

The odor recognition task involved each subject being presented with 10 common household odors to memorize, with a 30 sec interval between each presentation. After an interval of 15 min, a total of 20 odors were presented to subjects. This test set comprised of the 10 previously presented odors, and 10 new odors, presented in a random order. Subjects had to decide whether each odor was 'old' or 'new'. The signal detection data that resulted—although one or two of the counts might be out by one, because these data have been recovered from hit and false alarm rates truncated at two decimal places—are shown in Table 9.2.

Table 9.2: Recognition memory for odors reported by Lehrner et al. (1995).

	Control Group		Group I		Group II	
	Old Odor	New Odor	Old Odor	New Odor	Old Odor	New Odor
Old Resp.	148	29	150	40	150	51
New Resp.	32	151	30	140	40	139

Analyze these three data sets using signal detection theory to infer the discriminability and bias of the recognition performance for each group. What conclusions do you draw from this analysis? What, if anything, can you infer about individual differences between the subjects in the same groups?

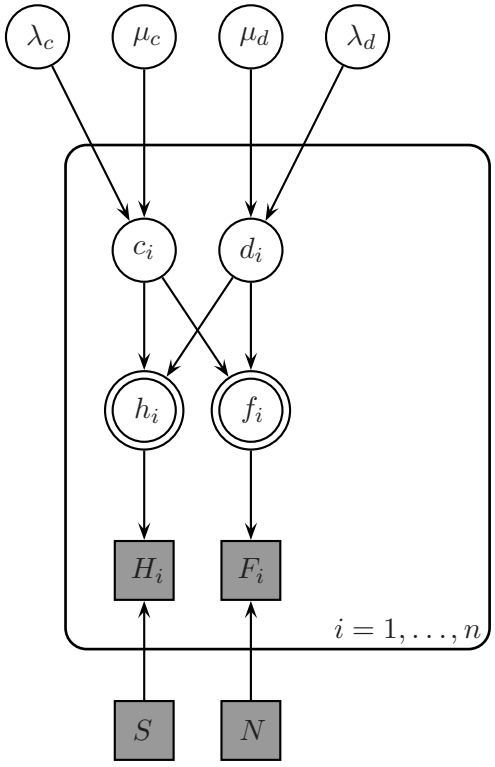
9.2 Hierarchical Signal Detection Theory

We now consider a hierarchical extension of SDT, applied to a different problem where individual subject data are available. This allows us to model possible individual differences using a hierarchical extension of the basic SDT model in the previous case study. The idea is that different subjects have different discriminabilities and biases that are drawn from group-level Gaussian distributions.

The data come from the empirical evaluation, presented by Heit and Rotello (2005), of a conjecture made by Rips (2001) that inductive and deductive reasoning can be unified within a signal detection theory framework. The conjecture involves considering the strength of an argument as a uni-dimensional construct, but allowing different criteria for induction and deduction. The criterion separates between 'weak' and 'strong' arguments in the inductive case, and 'invalid' and 'valid' arguments in the deductive case, with the deductive criterion being more extreme. Under this conception, deduction is simply a more stringent form of induction. Accordingly, empirical evidence for or against the SDT model has strong implications for the many-threaded contemporary debate over the existence of different kinds of reasoning systems or processes (e.g., Chater & Oaksford, 2000; Heit, 2000; Parsons & Osherson, 2001; Sloman, 1998).

In their study Heit and Rotello (2005) tested the inductive and deductive judgments of 80 participants on eight arguments. They used a between-subjects design, so that 40 subjects were asked induction questions about the arguments (i.e., whether the conclusion was "plausible"), while the other 40 participants were asked deduction questions (i.e., whether the conclusion was "necessarily true"). These decisions made by participants have a natural characterization in term of hit and false alarm counts.

One of the key analyses of Heit and Rotello (2005) used standard significance testing to reject the null hypothesis that there was no difference between discriminability for induction and deduction conditions. Their analysis involved calculating the mean discriminabilities for each participant, using edge-corrections where perfect performance was observed. These sets of discriminabilities gave means of 0.93 for the deduction condition and 1.68 for the induction condition. By calculating via the t statistic, and so assuming associated Gaussian sampling distributions, and observing that the p-value was less than .01, Heit and Rotello (2005) rejected the null hypothesis of equal means. According to Heit and Rotello (2005), this finding of different discriminabilities provided evidence against the criterion-shifting uni-dimensional account offered by SDT.



$\mu_d \sim \text{Gaussian}(0, .001)$

$\lambda_d \sim \text{Gamma}(.001, .001)$

$\mu_c \sim \text{Gaussian}(0, .001)$

$\lambda_c \sim \text{Gamma}(.001, .001)$

$d_i \sim \text{Gaussian}(\mu_d, \lambda_d)$

$c_i \sim \text{Gaussian}(\mu_c, \lambda_c)$

$h_i = \Phi(\frac{1}{2}d_i - c_i)$

$f_i = \Phi(-\frac{1}{2}d_i - c_i)$

$H_i \sim \text{Binomial}(h_i, S)$

$F_i \sim \text{Binomial}(f_i, N)$

Figure 9.4: Graphical model for hierarchical signal detection theory.

While the statistical inference methods used by Heit and Rotello (2005) are widely used and accepted, they explicitly or implicitly make a number of problematic assumptions that can be dealt with effectively using the Bayesian approach. First, the uncertainty about the discriminability of each individual is ignored, since it is represented by a single point estimate. Intuitively, making decisions corresponding, for example, to three hits and one false alarm is consistent, to varying degrees, with a range of possible hit and false-alarm rates, and hence, to varying degrees, with a range of discriminabilities. The Bayesian approach naturally represents this uncertainty by making prior assumptions about hit and false-alarm rates, and then using the evidence provided by the decisions to calculate posterior distributions. These posterior distributions are naturally mapped into posterior distributions for discriminability and bias according to SDT, which avoids the need for ad-hoc edge corrections. Perhaps most importantly, the statistical analysis undertaken by Heit and Rotello (2005) above (implicitly) assumes there are no individual differences across participants within each condition.

A graphical model for inferring discriminability and bias from hit and false alarm counts for $i = 1, \ldots, n$ subjects, but allowing Gaussian variation in the discriminability and bias across the group of subjects, is shown in Figure 9.4.

The following code implements the graphical model in WinBUGS.

```
# Hierarchical Signal Detection Theory
model{
   # Relating observed counts to underlying Hit and False Alarm rates
   for (i in 1:n) {
      HR[i] ~ dbin(h[i],S[i])
      FA[i] ~ dbin(f[i],N[i])
      S[i] <- HR[i]+MI[i]
      N[i] <- FA[i]+CR[i]
```

```
   }
   # Reparameterization Using SDT
   for (i in 1:n) {
         h[i] <- phi(d[i]/2-c[i])
         f[i] <- phi(-d[i]/2-c[i])
   }
   # Group Distributions
   for (i in 1:n) {
         c[i] ~ dnorm(muc,lambdac)
         d[i] ~ dnorm(mud,lambdad)
   }
   # Priors
   muc ~ dnorm(0,.001)
   mud ~ dnorm(0,.001)
   lambdac ~ dgamma(.001,.001)
   lambdad ~ dgamma(.001,.001)
   sigmac <- 1/sqrt(lambdac)
   sigmad <- 1/sqrt(lambdad)
}
```

The code `SDT_2.m` (Matlab) or `SDT_2.R` (R) applies the model to the Heit and Rotello (2005) data. Of key interest for testing the Rips (2001) conjecture is how the group-level means for bias and (especially) discriminability differ between the induction and deduction conditions.

With the initial sampling setting in this script, the joint posterior of these means will resemble that shown in Figure 9.5. The dark points in the joint distribution scatterplot and dark lines in the marginal densities correspond to the deduction condition, while the light points and lines correspond to the induction condition. Notice the strange set of samples leading from zero to the main part of the sampled distribution. This is clear evidence that the sampling run requires some samples before it locates itself in the posterior distribution. This problem is exactly what the concept of burn-in sampling was introduced to solve. Burn-in samples are initial samples that are not recorded, and not used in the analysis. Also notice that the marginal densities are poorly estimated, because there are not enough recorded samples.

Exercises

1. Study the lack of convergence more formally—run multiple chains and compute tests of convergence such as the $\hat{R}$ statistic.
2. The convergence problem can be fixed by introducing a burn-in period and increasing the number of samples. Make the required changes and run the adjusted Matlab/R script. The new results should resemble those in Figure 9.6. With this new and improved analysis, what conclusion do you draw about the Rips (2001) conjecture? How else could this modeling approach be used to address the question?
3. Are there still aspects of the MCMC chain that worry you? Confirm your suspicion by monitoring and assessing the behavior of the MCMC chains for the individual `c[i]` parameters.

9.3 Parameter Expansion★

with Dora Matzke

If you run the code that implements the hierarchical signal detection model discussed above, you will notice that the chains of the hierarchical variance parameter σ_c did not converge properly. As shown in Figure 9.7, the σ_c chains got repeatedly stuck near zero (i.e., at iterations 1400, 5000, 8500 and 9800 in the induction condition, and at iteration 2500 in the deduction condition) and it required several iterations

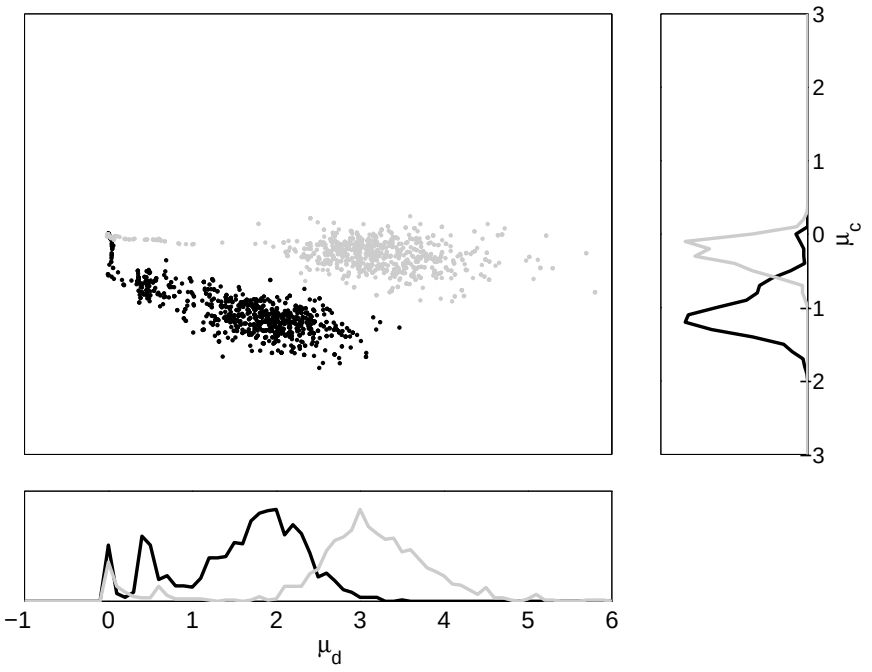


Figure 9.5: The joint posterior over μ_d and μ_c for the induction (dark) and deduction (light) conditions.

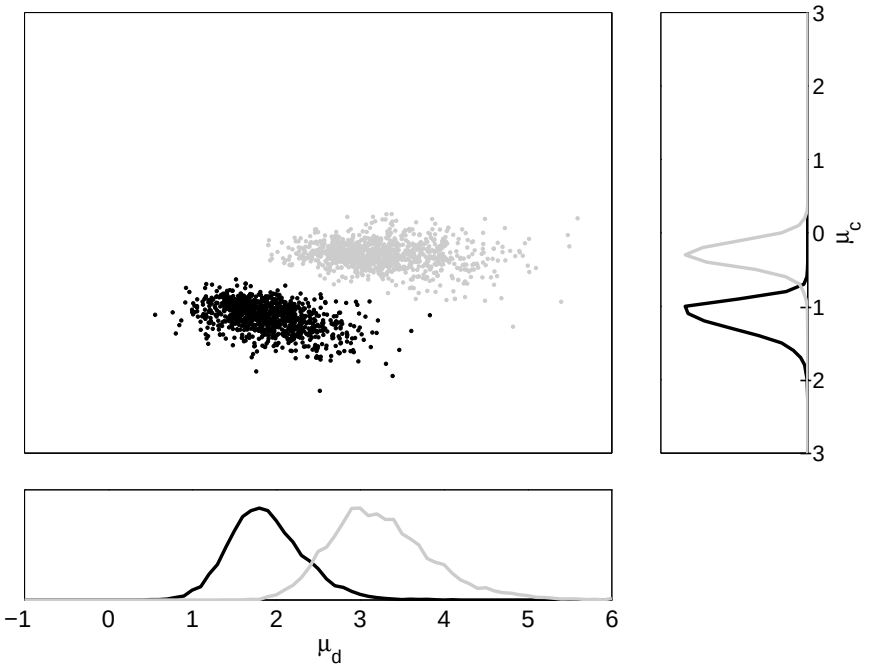
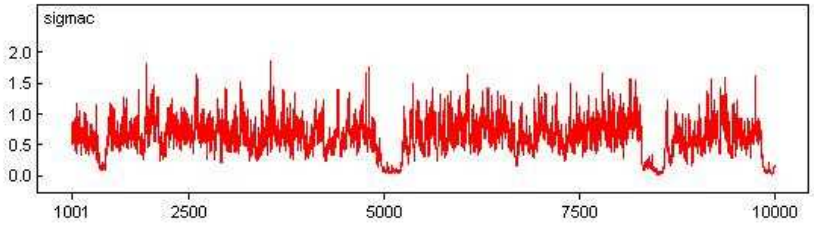
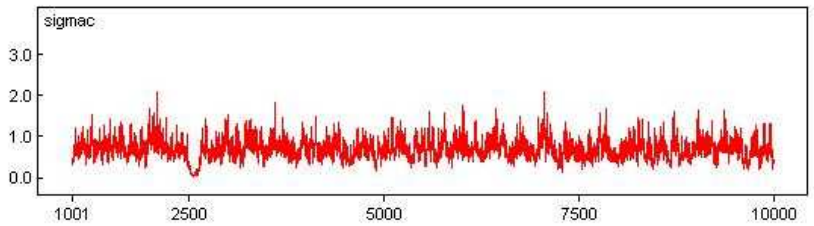


Figure 9.6: An improved joint posterior over μ_d and μ_c for the induction (dark) and deduction (light) conditions, using a burn-in period and more recorded samples.

before the chains moved forward. This particular type of nonconvergence is fairly common in complicated hierarchical Bayesian models. The problem is as follows. Suppose that σ_c happened to be estimated near zero. As a result, the bias c_i parameters will be pooled toward their population mean μ_c because the c_i's are updated based on the current value of σ_c. In turn, σ_c will be estimated near zero again because it is updated based on the current values of c_i. Eventually, the chain of σ_c will break out of the "zero variance trap". However, this may require several iterations and, as you can see in the top panel of Figure 9.7, the chain is likely to get stuck again.



Induction condition



Deduction condition

Figure 9.7: MCMC chains of the σ_c parameter of the hierarchical signal detection model.

A good way to enable the sampling process to escape the trap is to augment the original model with redundant multiplicative parameters using a technique known as parameter expansion (e.g., Gelman, 2004; Gelman & Hill, 2007; Liu & Wu, 1999). For example, we can extend our hierarchical signal detection model with two multiplicative parameters, say ξ_c and ξ_d. The role of these additional parameters is to rescale the original c_i and d_i parameters and their corresponding variances σ_c and σ_d.

The graphical model that implements the parameter-expanded model is shown in Figure 9.8. Note that the new model is equivalent to the original hierarchical signal detection model; the new model is simply a reparameterization of the original model. In the parameter-expanded model, c_i and σ_c are rescaled by multiplying them with ξ_c. The c_i parameter is now expressed in terms of μ_c, ξ_c and δ_{c_i}, and σ_c is expressed in terms of $|\xi_c|\sigma_c^{new}$. Similarly the d_i and σ_d parameters are rescaled by multiplying them with ξ_d. The d_i parameter is now expressed in terms of μ_d, ξ_d and δ_{d_i}, and σ_d is expressed in terms of $|\xi_d|\sigma_d^{new}$. The rationale behind parameter expansion is that updating the ξ_c and ξ_d parameters includes an additional random component in the sampling process. This component causes the samples of $\sigma_c = |\xi_c|\sigma_c^{new}$ and $\sigma_d = |\xi_d|\sigma_d^{new}$ to be less dependent on the previous iteration and prevents the chains to get trapped near zero regardless of how small their previous value was. Keep in mind, however, that in order to draw inferences under the original model, the parameters from the expanded model must be transformed back to their original scale. For example, if you want to draw inference about the original σ_c parameter, you should consider the estimate of $|\xi_c|\sigma_c^{new}$ instead of reporting only σ_c^{new}.

The following code implements the graphical model in WinBUGS.

```
# Hierarchical Signal Detection Theory
model{
   # Relating observed counts to underlying Hit and False Alarm rates
   for (i in 1:n) {
```

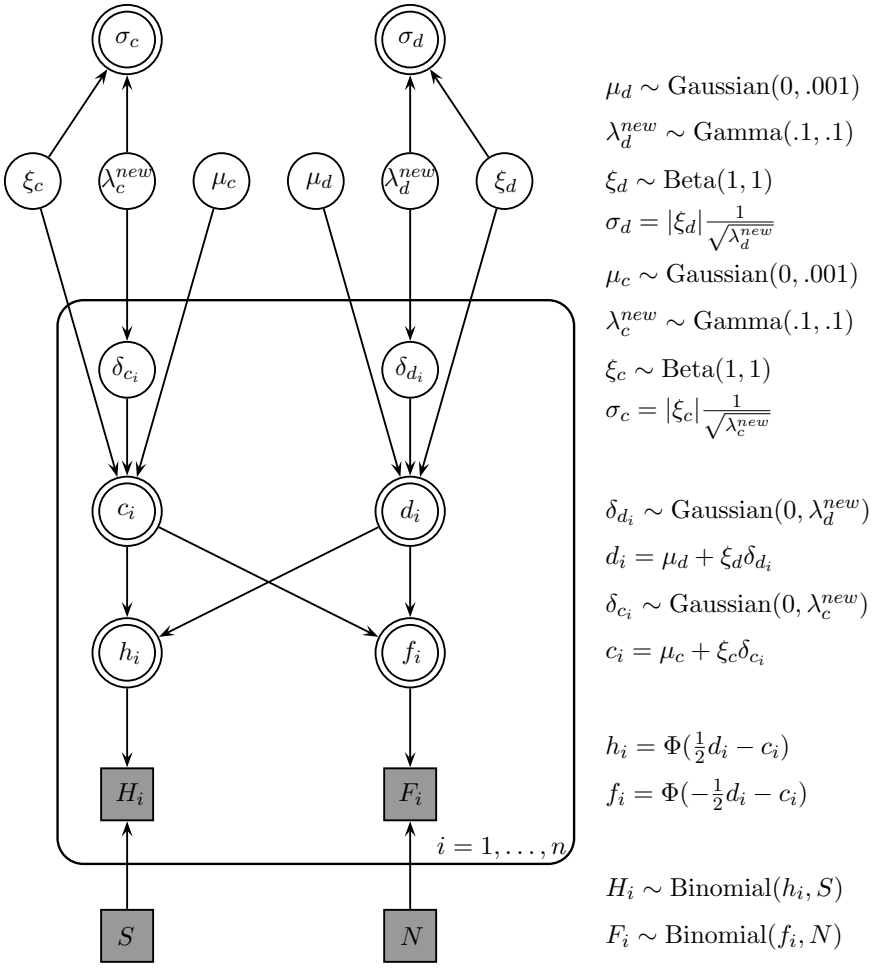
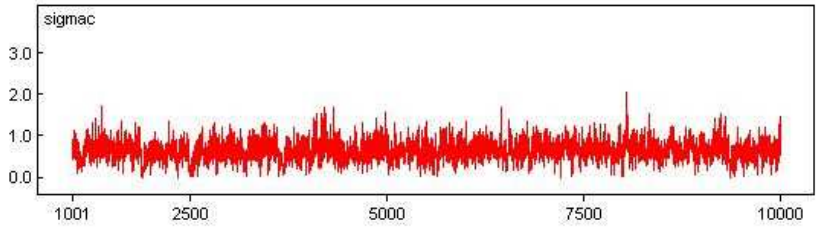


Figure 9.8: Graphical model for the parameter-expanded hierarchical signal detection theory.

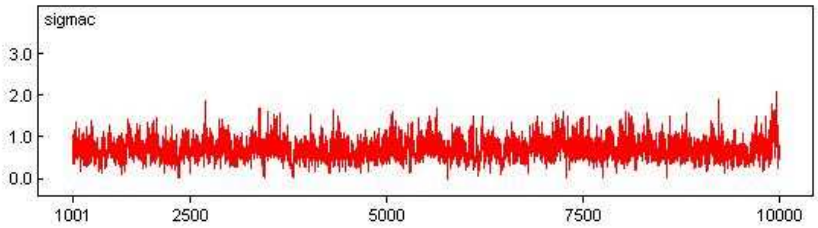
```
   HR[i] ~ dbin(h[i],S[i])
   FA[i] ~ dbin(f[i],N[i])
   S[i] <- HR[i]+MI[i]
   N[i] <- FA[i]+CR[i]
}
# Reparameterization Using SDT
for (i in 1:n) {
      h[i] <- phi(d[i]/2-c[i])
      f[i] <- phi(-d[i]/2-c[i])
}
# Group Distributions
for (i in 1:n) {
      c[i] <- muc+ xiC*deltac[i]
      d[i] <- mud+ xiD*deltad[i]
      deltac[i] ~ dnorm(0,lambdacNew)
      deltad[i] ~ dnorm(0,lambdadNew)
}
# Priors
```

```
   muc ~ dnorm(0,0.001)
   mud ~ dnorm(0,0.001)
   xiC ~ dbeta(1,1)
   xiD ~ dbeta(1,1)
   lambdacNew ~ dgamma(.1,.1)
   lambdadNew ~ dgamma(.1,.1)
   sigmacNew <- 1/sqrt(lambdacNew)
   sigmadNew <- 1/sqrt(lambdadNew)
   sigmac <- abs(xiC)*sigmacNew
   sigmad <- abs(xiD)*sigmadNew
}
```

The code `SDT_3.m` (Matlab) or `SDT_3.R` (R) applies the parameter-expanded model to the Heit and Rotello (2005) data. After you run the code, WinBUGS should display MCMC chains similar to those shown in Figure 9.9. The chains for the variance parameter σ_c now seem to have properly converged; the chains have successfully escaped the "zero variance trap" and look like fat hairy caterpillars.



Induction condition



Deduction condition

Figure 9.9: MCMC chains of the σ_c parameter of the parameter-expanded hierarchical signal detection model.

CHAPTER 10

MULTIDIMENSIONAL SCALING MODELS OF STIMULUS REPRESENTATION

Multidimensional Scaling (MDS) representations of stimuli use a low-dimensional metric space in which points correspond to stimuli (e.g., Shepard, 1980). The idea is to infer where these points should lie—based on similarity data that (one way or another) experimentally measure the similarities between every possible pair of stimuli—so that more similar stimuli have nearer points.

There are many sorts of MDS models, but we will just consider a simple but very useful one known sometimes as 'metric' MDS. To do this, we will assume the similarities are related to distances using the exponential decay relationship advocated by Shepard (1987). That is, we will assume that the empirically observed similarities decay exponentially as the distance between their representing points increases.

One interesting issue in MDS modeling involves the interpretation of different possible metrics used to measure the distances. Typically, consideration is restricted to the Minkowskian family of distance metrics. For points $\mathbf{p}_i = (p_{i1}, \ldots, p_{iD})$ and $\mathbf{p}_j = (p_{j1}, \ldots, p_{jD})$ in a D-dimensional space, the Minkowski r-metric distance is given by

$$d_{ij} = \left[\sum_{x=1}^{D} |p_{ix} - p_{jx}|^r \right]^{1/r} . \tag{10.1}$$

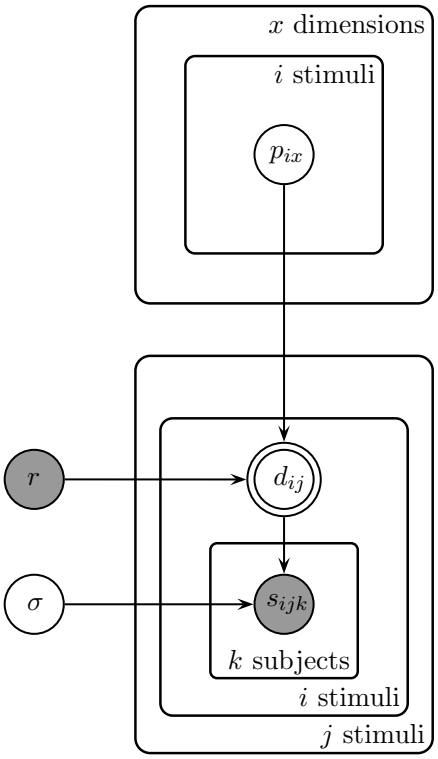
The $r = 1$ (City-Block) and $r = 2$ (Euclidean) cases are usually associated with, respectively, so-called 'separable' and 'integral' stimulus domains (Garner, 1974; Shepard, 1991). The basic idea is that many stimulus domains, like different shapes of different sizes, have component dimensions that can be attended to separately. These are termed separable, and are well modeled by the distance metric that treats each dimension independently in accruing distance. Other stimulus domains, like color, however, have component dimensions that are 'fused', and not easily distinguished, and so the comparison of stimuli involves all of the dimensions simultaneously. These are termed integral, and are well modeled by the familiar Euclidean distance metric. In addition, metrics with $r < 1$ have been given a psychological justification (e.g., Gati & Tversky, 1982; Shepard, 1987, 1991) in terms of modeling stimuli with component dimensions that 'compete' for attention.

10.1 City-Block MDS

Figure 10.1 presents a graphical model for MDS. At the top is the coordinate representation of the points corresponding to stimuli. The p_{ix} node corresponds to the single coordinate value of the ith stimulus on the xth dimension, and the surrounding plates repeat these coordinates over the stimuli and dimensions. We make the obvious prior assumption that all of the coordinates have equal prior probability of being anywhere in a large region. Given the value of r, and the coordinate locations p_{ix}, the pairwise distances d_{ij} are automatically given by Equation 10.1.

The similarity data we consider provide similarity ratings for each pair of stimuli as generated independently by many subjects. The observed similarity between the ith and jth stimuli given by the kth subject is denoted s_{ijk}, and so is enclosed by an additional plate representing the subjects. These similarities are assumed to be generated as the exponential decay of the distance between these points, but subject to Gaussian noise with common variance across all subjects and stimulus pairs. We put a uniform prior distribution on the standard deviation of the noise, as recommended by Gelman (2006).

This graphical model is implemented by the following WinBUGS code. Notice the way the `pt` variables are mapped to a translation invariant form `p`. This is because if all the coordinate points in an MDS representation are moved in the same direction, the distances between the points are preserved. To remove this indeterminacy, the `p` variable at each sample is `pt` variable with the center of mass constrained to be the origin.



$$p_{ix} \sim \text{Uniform}(-5, 5)$$

$$d_{ij} = \left[\sum_{x=1}^{K} |p_{ix} - p_{jx}|^r\right]^{1/r}$$

$$s_{ijk} \sim \text{Gaussian}(\exp(-d_{ij}), 1/\sigma^2)$$

$$\sigma \sim \text{Uniform}(0, 0.5)$$

Figure 10.1: Graphical model for metric multidimensional scaling.

```
# Multidimensional Scaling
model {
    # Similarity data
    for (i in 1:NSTIM) {
       for (j in 1:NSTIM) {
          for (k in 1:NSUBJ) {
              s[i,j,k] ~ dnorm(mu[i,j],lambda)I(0,1)
          }
          mu[i,j] <- exp(-d[i,j])
          d[i,j] <- pow(pow(abs(pt[i,1]-pt[j,1]),r)+pow(abs(pt[i,2]-pt[j,2]),r),1/r)
       }
    }
    # Translation Invariant Points
    for (i in 1:NSTIM) {
       for (x in 1:2) {
          p[i,x] <- pt[i,x]-mean(pt[1:NSTIM,x])
       }
    }
    # Priors
    for (i in 1:NSTIM) {
       for (x in 1:2) {
          pt[i,x] ~ dunif(-5,5)
       }
    }
    sigma ~ dunif(0,0.5)
```

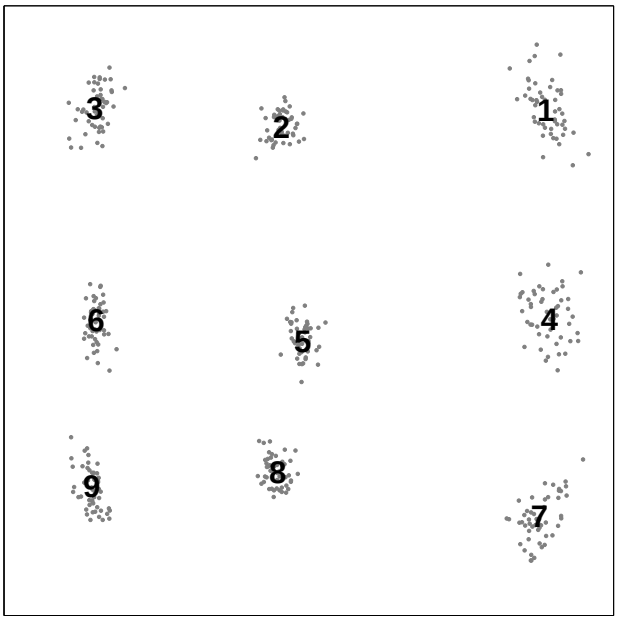


Figure 10.2: Multidimensional scaling points posterior representation.

```
    lambda <- 1/pow(sigma,2)
}
```

The code `MultidimensionalScaling_1.m` (Matlab) or `MultidimensionalScaling_1.R` (R) applies the model to data giving similarity ratings for 9 participants on 9 circles of different sizes with radial lines at different angles, following (essentially) a 3×3 factorial design, as reported by Treat, MacKay, and Nosofsky (1999). The analysis assumes the stimuli are separable, and so uses $r = 1$.

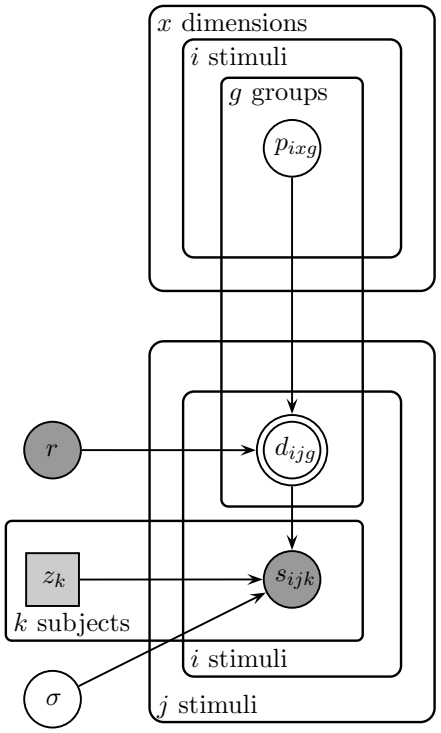
The posterior distribution of the representing points is shown in Figure 10.2, and clearly shows the basic 3×3 design. Notice how the posterior naturally expresses the uncertainty about the exact representing point assumed by the MDS representational model.

Exercises

1. Draw the posterior distribution for the common variance.
2. Do some analysis that presents an account of how well the MDS representation fits the data. There are many possibilities. One way would be to show the relationship between the `mu` variables and the empirical similarities, perhaps as a scatter-plot. Another way might be to overlay the empirical distances on a representational display like Figure 10.2. There are certainly other ways: Be creative.

10.2 Euclidean MDS With Individual Differences

Our second MDS analysis relates to the similarities between ten spectral colors, as reported by Helm (1959). Color is regarded as the prototypical integral stimulus domain, and so provides a contrast with the separable domain we have just used. And spectral color MDS representations are a famous example of the whole MDS idea, because they typically transform a physical representation based on the wavelength single dimension into a two-dimensional psychological representation, known as a 'color circle' or 'color horse-shoe'. In this representation, the high psychological similarity between red and violet, at opposite ends of the physical spectrum, is represented.



$$p_{ixg} \sim \text{Uniform}(-5, 5)$$

$$d_{ijg} = \left[\sum_{x=1}^{K} |p_{ixg} - p_{jxg}|^r\right]^{1/r}$$

$$s_{ijk} \sim \text{Gaussian}(\exp(-d_{ijz_k}), 1/\sigma^2)$$

$$z_k \sim \text{Bernoulli}(1/2)$$

$$\sigma \sim \text{Uniform}(0, 0.5)$$

Figure 10.3: Graphical model for metric multidimensional scaling with individual differences.

The Helm (1959) data have the additional interesting feature of containing individual differences. Of the 16 subjects, 11 are known to have normal color vision, while 5 have a color deficiency known as deuteronomy.

Figure 10.1 presents a graphical model for MDS that provides one (but certainly not the only) way to express and make inferences about the individual differences. The basic idea is that there are now two groups of subjects, and each group has their own MDS representation. The binary variable z_k identifies to which group the k subject belongs. In our analysis, we will treat some of these assignments as known (i.e., observed), and some as unknown (i.e., latent). The terms 'semi-supervised' and 'partially observed' are good ones to describe this set-up, and the lighter shading, between the fully observed and unobserved shadings, is intended to indicate the idea in the graphical model.

The following WinBUGS code implements the graphical model in Figure 10.3 in the context of a particular analysis, where the assignment of 10 subjects is known (the `z` variables that are assigned values) but the assignment of the other 6 subjects is unknown (the `z` variables that are assigned priors).

```
# Multidimensional Scaling With Two Groups
model {
    # Similarity data
    for (i in 1:NSTIM) {
       for (j in 1:NSTIM) {
          for (k in 1:NSUBJ) {
              s[i,j,k] ~ dnorm(mu[i,j,z[k]],lambda)I(0,1)
          }
          for (g in 1:2) {
              mu[i,j,g] <- exp(-d[i,j,g])
              d[i,j,g] <- pow(pow(abs(pt[i,1,g]-pt[j,1,g]),r)
              +pow(abs(pt[i,2,g]-pt[j,2,g]),r),1/r)
          }
```

```
      }
   }
   # Two Groups
   for (i in 1:6) {
      z01[i] ~ dbern(0.5)
      z[i] <- z01[i]+1
   }
   z[7]  <- 1
   z[8]  <- 1
   z[9]  <- 1
   z[10] <- 1
   z[11] <- 1
   z[12] <- 1
   z[13] <- 2
   z[14] <- 2
   z[15] <- 2
   z[16] <- 2
   # Translation Invariant Points
   for (i in 1:NSTIM) {
      for (x in 1:2) {
         for (g in 1:2) {
            p[i,x,g] <- pt[i,x,g]-mean(pt[1:NSTIM,x,g])
         }
      }
   }
   # Priors
   for (i in 1:NSTIM) {
      for (x in 1:2) {
         for (g in 1:2) {
            pt[i,x,g] ~ dunif(-5,5)
         }
      }
   }
   sigma ~ dunif(0,0.5)
   lambda <- 1/pow(sigma,2)
}
```

The code `MultidimensionalScaling_2.m` (Matlab) or `MultidimensionalScaling_2.R` (R) completes the analysis, choosing the 10 known and 6 unknown assignments from the Helm (1959) data. Notice that the metric parameter is set to $r = 1.8$, which is near integral, but not fully integral. This is consistent with the results for these data presented by Lee (2008), who made inferences about the metric parameter. It also sidesteps a weakness of the current WinBUGS implementation of the MDS model, which has translation but not rotation invariance. For $r \neq 2$ there is no natural rotation invariance, but this would be needed if $r = 2$ was assumed. I think the right way to approach this is, much as the center of mass was used to solve the translation problem, to use the moment of inertia (the rotational mechanics equivalent of center of mass), and rotate to the principal axis representation. But I do not know how to do this yet.

A simple analysis of the unobserved `z` variables shows that the correct inferences are made. The first 4 subjects are really color-normal, the final two are color-deficient. The posterior distribution of the representing points for both groups are shown in Figure 10.2, and clearly shows the perfect color circle for the color-normal group, and the distorted color-circle for the deuteronomy group.

Exercises

1. Assess whether or not the MCMC chains for the different parameters have converged.

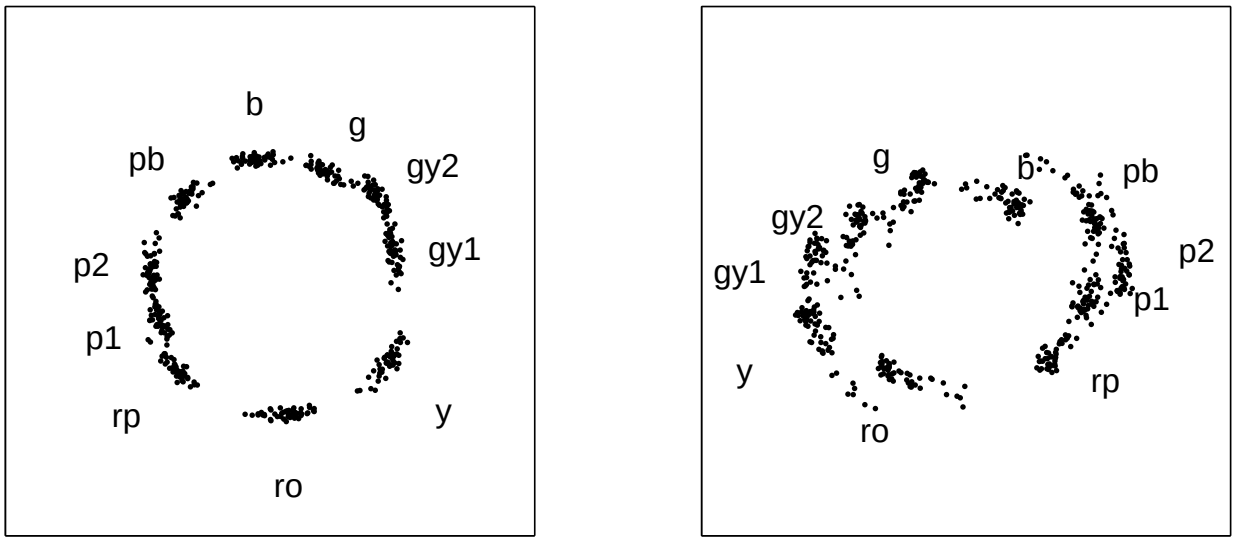


Figure 10.4: Multidimensional scaling points posterior representations for the normal (left panel) and deficient (right panel) sub-group.

2. What do you think of partial observability? Does it depend on how many observed cases are needed to make good inferences about the unobserved ones? (Compare the situation where for every 6 undiagnosed subjects we needed 10 diagnosed ones, versus the case where once we have 10 diagnoses, every new subject can be correctly diagnosed using this approach).
3. It would be possible to write much more elegant and general code for the partial observability. Think about a good way to achieve this.
4. You could solve the rotation invariance issue for me.

CHAPTER 11

THE TAKE THE BEST MODEL OF HEURISTIC DECISION-MAKING★

The Take the Best (TTB) heuristic model of decision-making (Gigerenzer & Goldstein, 1996) is a very simple account of how people choose between two stimuli on some criterion. It addresses decision tasks like "which of Frankfurt or Munich has the larger population?", "which of a catfish and a herring is more fertile?", and "which of these two professors has the higher salary?". These are clearly all important issues.

TTB assumes that all stimuli are represented in terms of the presence or absence of a common set of cues. So, Frankfurt and Munich, along with other German cities, might be represented in terms of whether or not they have an international airport, whether they have hosted the Olympics, whether they have a football/soccer team in the Bundesliga, and so on. TTB assumes these cues are always binary, says nothing about how they are generated, nor how they are assigned to stimuli, nor how they and their assignments change over time. Remember the day you were in St. Pauli when Hamburg were relegated from the Bundesliga?

Associated with each cue in TTB is a 'cue validity'. This validity measures the proportion of times, for those pairs of stimuli where one has the cue and the other does not, that the cue belonged to the stimulus that with the greater criterion value. For example, the validity of the cue 'has an airport', for a set of cities, is the proportion of times the larger city has the airport, when attention is restricted to just those pairs of cities where only one has an airport. By this definition, cue validities must range between 0 and 1. In practice, if a cue validity was smaller than 0.5, it makes sense to recode the cue in terms of the absence of the property. This tactic guarantees all validities are between 0.5 and 1. TTB does not explicitly say much about how validities are learned from experience, although Gigerenzer and colleagues have presented some relevant work (e.g., Todd & Dieckmann, 2005; Martignon & Laskey, 1999).

TTB as a cognitive process account of how people make decisions is based around interesting and sensible answers to three basic questions.

- The first question is "how do people search their environment for information?". TTB says people search the cues in validity order, starting with the highest validity cue and working their way down.
- The second question is "how do people decide to terminate their search for information?". Here TTB proposes a radical answer, saying that people stop searching as soon as they have found one discriminating cue. That is, the first cue found that belongs to one stimulus but not the other is enough to stop searching.
- The final question is "once people have stopped searching, how do they decide?". TTB proposes, simply, that people choose the stimulus with the first discriminating cue. If no discriminating cue is found, and all of the cues are exhausted, TTB proposes people guess.

11.1 TTB With Fixed Search Order

A graphical model for implementing TTB is shown in Figure 11.1. The experimental data are the decisions made for a set of questions. The binary k_q indicates the decision (i.e., the stimulus that was chosen) for the qth question, and p_{q1} and p_{q2} give the two stimuli for qth question. The patterns of cue membership are also known, with $c_{ik} = 1$ if the ith stimulus has the kth cue, and $c_{ik} = 0$ if it does not. The search order for cues is specified by the observed variable s.

To make the inherently deterministic TTB model a probabilistic model suited to Bayesian implementation, a responding rate parameter γ is included. The idea is that the model follows the decision made by TTB with probability γ. This is done via the deterministic θ_q variable, which, for every qth question, gives probability γ to the TTB choice and $1 - \gamma$ to the non-TTB choice. We use a $\gamma \sim \text{Beta}(1, 1)$ prior.

The following code implements TTB in WinBUGS. The code assumes that the cues representing stimuli are in the desired search order. So, if cues are to be searched in order of their validities, as in TTB, the first

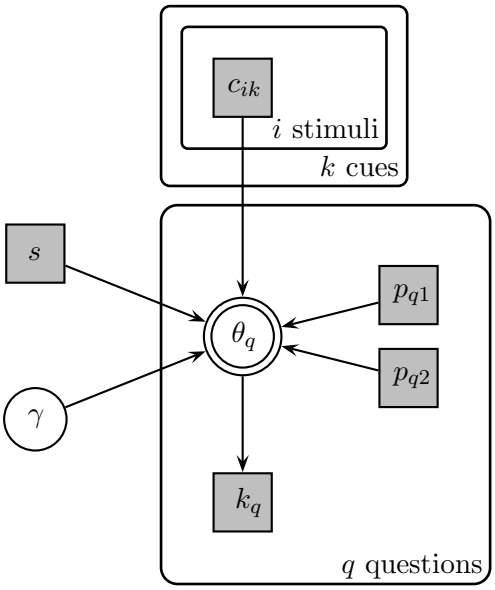


Figure 11.1: Graphical model for TTB with known search order.

cue needs to be the most valid, the second cue the second-most valid, and so on (i.e., the information in s in the graphical model is implicitly in the c_{ik} values). One advantage of this approach is that other fixed search strategies could easily be implemented by supplying the cues in different orders.

There are a couple of noteworthy things in the code. The basic approach is to calculate a weighted sum for each question (i.e., each stimulus pair), with more valid cues receiving exponentially more weight. This means the total weight is dominated by the highest validity cue for each stimulus, and the stimulus with the highest validity cue overall will have the greatest weighted sum.

The other trick is more of an ugly hack. WinBUGS will not sample the deterministic variable `sc` that measures the accuracy of TTB. To overcome this, the stochastic variable `gamma` is added and subtracted in defining `sc`. This makes WinBUGS happy to record the samples for `sc`, without (obviously) changing its value.

```
# Take The Best With Known Search Order
model {
   # Observed Decisions
   for (i in 1:nq) {
      k[i] ~ dbern(theta[i,d1[i]])
   }
   # TTB Decision
   for (i in 1:nq){
      # Cue contributions to decision
      for (j in 1:nc){
         w[i,j] <- (c[p[i,1],j]-c[p[i,2],j])*pow(2,vo[j]-1)
      }
      # TTB takes the first cue, or they are not different
      # First object gives d1[i]=3, Second object gives d1[i]=1, Same gives d1[i]=2
      dtmp[i] <- sum(w[i,1:nc])
      d[i] <- -1*step(-dtmp[i])+step(dtmp[i])
      d1[i] <- d[i]+2
      # Goes with TTB decision with probability gamma, or guesses
      theta[i,1] <- 1-gamma
```

```
      theta[i,2] <- 0.5
      theta[i,3] <- gamma
      # Whether decision was correct
      correct[i] <- equals(k[i],1)*equals(d1[i],3)
                    +equals(k[i],0)*equals(d1[i],1)
                    +0.5*equals(d1[i],2)
   }
   # Proportion of correct decision, with hack to insure monitoring
   sc <- sum(correct[1:nq])/nq+gamma-gamma
   # Priors
   gamma ~ dbeta(1,1)
}
```

The code `TakeTheBest_1.m` (Matlab) or `TakeTheBest_1.R` (R) uses two data sets made available by the Gigerenzer group. The `Citworld` one is the famous German cities data set, which has 83 cities represented in terms of 9 cues, with population as the criterion. The professor data sets is less well studied (and less well documented), but has fewer stimuli and cues, and so is a computationally less burdensome first example.

The Matlab/R code generates the observed 'experimental' data by finding the correct decision for each pair of stimuli. That is, it behaves as if it were a subject in a task asked to compare every possible pair of stimuli, and always makes the correct decision. In the context of a real experiment, this part of the could would be replaced with experimental data giving the decisions subjects actually made, for whatever sequence of questions they were asked.

Running this code will give as the main output the variable `sc`, which is simply the proportion of the observed decisions correctly predicted by TTB. (Note that because this variable is deterministic, WinBUGS will crash trying to draw its density. Look at its history or stats instead in the GUI).

Exercises

1. Try a few different search orders, by changing the Matlab/R code to alter the order of the columns of the `c` matrix. What effect does this have on the accuracy of TTB? Is the cue validity order recommended by TTB guaranteed to be the best order? If so, why? If not, why not?

2. The original exposition of TTB (Gigerenzer & Goldstein, 1996) included a 'recognition' stage that preceded the search of cues. The idea is that if one stimulus is recognized but the other is not (e.g., you have heard of Berlin but not Magdeburg), you will choose the one you recognize. If neither stimulus is recognized, TTB says you guess. If both are recognized, the cue search process we have implemented is used. How could the current graphical model be extended to include this recognition stage?

3. What is the relationship between the γ responding rate and the `sc` accuracy measure?

11.2 Inferring the TTB Search Order

One of the nice consequences of representing cognitive models in the form of graphical models is that alternative analyses often suggest themselves immediately. A good example of this relates to the search order. Figure 11.2 presents a modified graphical model in which the search order `s` is now unobserved.

This small modeling change makes the psychology of the inference much more interesting. Now, rather than assuming a search order, an inference is made about what search order is being used, based on the observed decisions being made. This means, of course, that the search order variable `s` needs a prior, and we assume that all possible search orders (for m cues there are $m!$ possible search orders) are equally likely.

The following code gives the WinBUGS implementation of the graphical model in Figure 11.2. Implementing the sampling of search orders is achieved in an interesting way. The simplest possible would be to have a categorical variable that could take $m!$ values, each corresponding to a search order. But doing inference on this variable would hampered by the lack of structure in the relationships between the values. If two search orders are similar (say they have the same order, except for two neighboring cues, whose orders

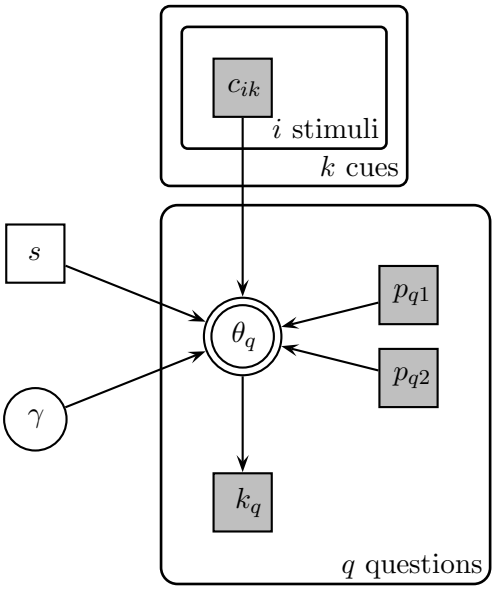


Figure 11.2: Graphical model for TTB with unknown search order.

are flipped) it would help if they had similar (i.e., neighboring) categorical numbers. But it is obviously impossible to map all the structure relationships between the orders onto the number line.

Instead, our approach uses separate, but carefully constrained, underlying categorical variables for each of the cues, and their values on these variables are ranked to derive the cue order. For a domain with m cues, the first cue can take values $1, \ldots, m{+}1, \ldots, 2m{+}1, \ldots$, the second cue can values $2, \ldots, m{+}2, \ldots, 2m{+}2, \ldots$, and so on. This means that small changes in the underlying cue-specific categorical variables will lead to small changes in the search order, and sampling works much better.

```
# Take The Best With Unknown Search Order
model {
   # Observed Decisions
   for (i in 1:nq) {
      k[i] ~ dbern(theta[i,d1[i]])
   }
   # TTB Decision
   for (i in 1:nq){
      # Cue contributions to decision
      for (j in 1:nc){
         w[i,j] <- (c[p[i,1],j]-c[p[i,2],j])*pow(2,cvo[j]-1)
      }
      # TTB takes the first cue, or they are not different
      # First object gives d1[i]=3, Second object gives d1[i]=1, Same gives d1[i]=2
      dtmp[i] <- sum(w[i,1:nc])
      d[i] <- -1*step(-dtmp[i])+step(dtmp[i])
      d1[i] <- d[i]+2
      # Goes with TTB decision with probability gamma, or guesses
      theta[i,1] <- 1-gamma
      theta[i,2] <- 0.5
      theta[i,3] <- gamma
      # Whether decision was correct
      correct[i] <- equals(k[i],1)*equals(d1[i],3)
```

```
                    +equals(k[i],0)*equals(d1[i],1)
                    +0.5*equals(d1[i],2)
   }
   # Proportion of correct decision, with hack to insure monitoring
   sc <- sum(correct[1:nq])/nq+gamma-gamma
   # Cue Search Order
   for (i in 1:nc){
      cvo[i] <- rank(cv[1:nc],i)
   }
   # Priors
   gamma ~ dbeta(1,1)
   for (i in 1:nc){
    cv[i] ~ dcat(base[i,1:M])
   }
}
```

The code `TakeTheBest_2.m` (Matlab) or `TakeTheBest_2.R` (R) uses the same data sets considered earlier, and then calls WinBUGS to apply the graphical model from which search order can be inferred. The Matlab/R code generates the `base` variables that give the priors for the underlying categorical variables for each cue, so that they can only sample the constrained sequence of values required to define the search orders.

Running the code will produce a simple analysis of the posterior for the search order, listing how many different search orders were sampled, what those orders were, their estimated posterior mass (i.e., what proportion of times each order was sampled), and the accuracy of each order, as measured by the `sc` variable.

Exercises

1. Plot the posterior density of `sc`.
2. Change the WinBUGS script so that the `theta` variables are cut, and so the observed decisions no longer cause any inference in the model. You will need to change the line giving the data distribution to `k[i]` $\sim$ `dbern(theta.cut[i,d1[i]])`, and also insert lines something like `theta.cut[i,1] <- cut(theta[i,1])` for all three possibilities. Run this model and plot the posterior density of `sc` again. What is the relationship between the two posterior densities you have produced? Explain the different posteriors over search orders when `theta` is cut and not cut. Most importantly, what does all this mean psychologically?

CHAPTER 12

CHILDREN'S DEVELOPMENT OF NUMBER CONCEPTS

This example involves two competing theories of how children develop concepts for number words. One theory—known as the 'knower level' theory—asserts that children learn the meaning of the number words in order. The claim is that children understanding the meaning of 'one' first, then 'two', then 'three', and then (usually) 'four', at which point they work out the pattern, and understand the meaning of all counting numbers. In the jargon of the theory, children start being '0-knowers', progress to '1-knowers' once they understand 'one', though '2-knower', '3-knower' and (usually) '4-knower' levels, until then eventually become 'HN-knowers' (for "Higher Number").

The other theory—known as the 'analog representation' theory—assumes that children, as with animals, have an innate analog representation system for numbers that they must learn to map onto the number words. The idea is that this mapping works like other psychophysical mappings, so that it is approximately correct, but with variability in performance that increases as the number being estimated grows larger.

Despite their quite different representational assumptions, both theories make similar predictions for many standard developmental tasks and research questions. Both predict approximately accurate answers to number-based questions with improvement as children age. There is, however, at least one important developmental task—known as the 'give-n' task—where the two theories make different predictions. In a give-n task, children are simply asked to give some number of objects (usually toys) to the experimental (or an experimenter substitute, like a puppet). The behavioral data are just a set of question-answer pairs, recording how many the child was asked to give and how many they actually gave.

The divergent predictions of the two theories on the give-n task relate to errors, particularly in terms of over- versus under-estimation. The analog representation theory predicts that errors will be symmetrical, because they arise from mis-estimation, and so it is just as likely a child asked to give two toys will incorrectly give one as that they will give three. The knower level theory, on the other hand, predicts that errors will usually be over-estimations, because children learn the number words in sequence. So, if a child understand the number one (i.e., is a "1-knower") they might mistakenly give three toys when asked for two, but they will not give one toy (because they know what "one" means).

If this were all there were to the story, it would be easy to do an analysis of the raw behavioral data to test the knower-level and analog representation theories, just by looking at the patterns of errors. Two things (at least) make life more complicated. The first is that the knower-level predictions depend heavily on whether a child is a 0-knower, 1-knower, and so on, (e.g., a 2-knower might well give three when asked for four, but a 3- or 4- or HN-knower should not), so this needs to be inferred.

The second big complicating issue is that there are task-specific influences on behavior. It is empirically quite likely a 0-knower, whatever they are asked for, will give one toy, or two toys, or a small handful of toys, or the whole basket-full of them. So, if the basket of toys the child selects from has 15 toys in total, answers like 1, 2, 3, and 15 are more likely than (say) 8, 10 or 12, but part of this is really just a task specific quirk.

To understand the data, then, and especially to evaluate the two competing representations theories using the data, we need a complete model of how children perform on the give-n task. That is, we need not just a theory of how they represent the numbers, but also how they make decisions on the task.

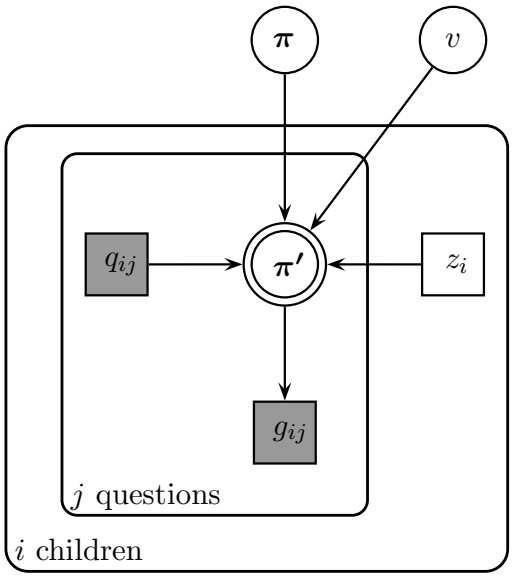
One simple way of building such models is to assume there are three parts to the decision-making of children in the give-n task.

- Initially, they have a 'base-rate' set of probabilities of giving each possible number of toys. This base-rate corresponds to something like the number they would spontaneously give, if there were not instructed to give a specific number.
- When the instruction is given, this acts as information to update the base-rate probabilities. How the updating happens will depend critically on how the numbers are represented (i.e., whether children use knower-level or analog representations), and the details of that representation (e.g., what their knower level is, or how variable their estimation accuracy is).

- Finally, the child will select a number to give based on the updated probabilities. At this point, they will produce the actual behavior observed in the task, and recorded by the data.

Fleshing out this framework for both the knower level and analog representation theories lets us analyze give-n data, and what it tells us about the theories.

12.1 Knower Level Model



$$\boldsymbol{\pi} \sim \text{Dirichlet}(1, \ldots, 1)$$

$$v \sim \text{Uniform}(1, 1000)$$

$$z_i \sim \text{Categorical}(\tfrac{1}{6}, \ldots, \tfrac{1}{6})$$

$$\pi'_{ijk} \propto \begin{cases} \pi_k & \text{if } q > z \\ v \times \pi_{ijk} & \text{if } q_{ij} \leq z_i \text{ and } k = q_{ij} \\ \frac{1}{v} \times \pi_{ijk} & \text{if } q_{ij} \leq z_i \text{ and } k \neq q_{ij} \end{cases}$$

$$g_{ij} \sim \text{Categorical}(\boldsymbol{\pi'})$$

Figure 12.1: Graphical model for behavior on the give-n task according to the knower level theory.

Figure 12.1 presents a graphical model for the knower level based account of how the give-n data are generated. The data are the observed q_{ij} and g_{ij} which give the number asked for (the 'question') and the answer (the number 'given') respectively, for the ith child on their jth question. The base-rate probabilities are representing by the vector $\boldsymbol{\pi}$, which is updated to $\boldsymbol{\pi'}$, from which the number given is sampled. The updating occurs using the number asked for, the knower level z_i of the child, and an evidence value v that measures the strength of the updating.

The logic of the updating can be explained using two cases. If the child is asked to give a number they know about then the probability of giving that number increases by a factor of v, and the probability of giving other numbers they know about but were not asked to give all decrease by the same factor v. The relative probabilities of giving other numbers are not changed, because the child does not know about them. So, for example, if a 3-knower is asked to give two, their probability of giving two increases, and their probability of giving one or three decreases, relative to the base rate. But the numbers four and above do not change in their relative probabilities, although they will in their absolute probabilities, because the probabilities for one, two and three have changed (i.e., if four was twice as likely as five in the base-rate, it still will be after updating, although both might be less likely in absolute terms).

The other case to consider is when the child is asked to give a number they do not know. Here, the numbers they do know will be updated to be less probable by the factor v, but all of the other numbers will retain the same relative probabilities to each other. So, for example, is a 3-knower is asked to give five, they become much less likely to give one, two or three, but equally relatively likely to give four and above, according to their base-rate.

The details of the definition of $\boldsymbol{\pi'}$ capture all of this logic, and it is worth thinking this through, to make sure you understand. The following script implements this graphical model in WinBUGS. There is quite

a bit going on here, and it is worth studying. Notice the way the conditional definition and normalization of the `piprime` probabilities are managed. Notice also that there are posterior predictions being made for each individual subject, as well as for each knower level as a group.

```
# Number Concept Using Knower Levels
model {
   # Data
   for (i in 1:S){
      for (j in 1:Q[i]){
         # Probability a z[i]-knower will give g[i,j] when asked for q[i,j]
         # is a categorical draw from their distribution over the 1:N toys
         g[i,j] ~ dcat(npiprime[z[i],q[i,j],1:N])
      }
   }
   # Likelihood
   for (i in 1:Z){
      for (j in 1:N){
         for (k in 1:N){
            piprimetmp[i,j,k,1] <- pi[k]
            piprimetmp[i,j,k,2] <- 1/v*pi[k]
            piprimetmp[i,j,k,3] <- v*pi[k]
            # Will be 1 if knower-level (i.e, i-1) is same or greater than answer
            ind1[i,j,k] <- step((i-1)-k)
            # Will be 1 for the possible answer that matches the question
            ind2[i,j,k] <- equals(k,j)
            # Will be 1 for 0-knowers
            ind3[i,j,k] <- equals(i,1)
            # Will be 1 for HN-knowers
            ind4[i,j,k] <- equals(i,Z)
            ind5[i,j,k] <- ind3[i,j,k]+ind4[i,j,k]*(2+ind2[i,j,k])
                           +(1-ind4[i,j,k])*(1-ind3[i,j,k])
                           *(ind1[i,j,k]+ind1[i,j,k]*ind2[i,j,k]+1)
            piprime[i,j,k] <- piprimetmp[i,j,k,ind5[i,j,k]]
            npiprime[i,j,k] <- piprime[i,j,k]/sum(piprime[i,j,1:N])
         }
      }
   }
   # Posterior Prediction For Knower Levels
   for (i in 1:Z){
      for (j in 1:N){
         for (k in 1:N){
            npiprime.cut[i,j,k] <- cut(npiprime[i,j,k])
         }
         ppz[i,j] ~ dcat(npiprime.cut[i,j,1:N])
      }
   }
   # Posterior Prediction For Subjects
   for (i in 1:S){
      z.cut[i] <- cut(z[i])
      for (j in 1:N){
         pp[i,j] ~ dcat(npiprime.cut[z.cut[i],j,1:N])
      }
   }
   # Baserate
```

```
   for (i in 1:N){
      pitmp[i] ~ dunif(0,1)
      pi[i] <- pitmp[i]/sum(pitmp[1:N])
      pi.cut[i] <- cut(pi[i])
   }
   ppb ~ dcat(pi.cut[1:N])
   # Priors
   v ~ dunif(1,1000)
   for (i in 1:S) {
      z[i] ~ dcat(priorz[])
   }
   priorz[1] <- 1/6 # 0-knower
   priorz[2] <- 1/6 # 1-knower
   priorz[3] <- 1/6 # 2-knower
   priorz[4] <- 1/6 # 3-knower
   priorz[5] <- 1/6 # 4-knower
   priorz[6] <- 1/6 # HN-knower
}
```

The code `NumberConcept_1.m` (Matlab) or `NumberConcept_1.R` (R) applies the model to data provided by Barbara Sarnecka. This code produces several analyses. Figure 12.2 shows the base-rate that is inferred. It shows an intuitively appealing result, giving high probability to small numbers of toys, as well as to the whole basketful. It is important to understand this was a latent multi-dimensional variable that was inferred from the data, and not something that was assumed to explain the data. This is a very powerful piece of inference that highlights a big advantage of the Bayesian approach we are adopting. I think it would be very hard to make this inference using 'standard' methods.

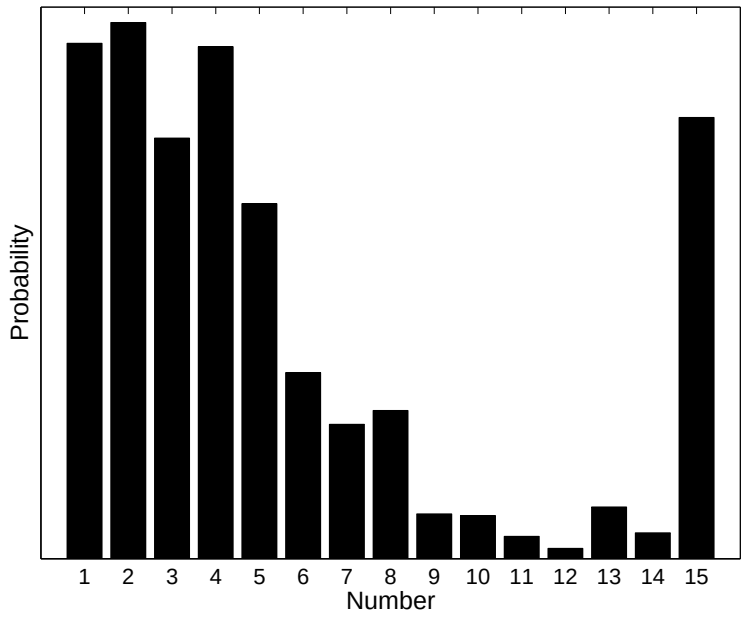


Figure 12.2: Posterior base rate for giving $1, \ldots, 15$ toys using the knower level model.

Figure 12.3 shows the posterior distribution over the six knower-level (0, 1-, 2-, 3-, 4- and HN-knowers) for each subject, ordered from the smallest expected value to the largest. The noteworthy feature of this result is that most of the subjects are classified with high certainty into a single knower level. There are exceptions (e.g., subjects 78, 45, ...), but, for the most part, there is confidence in a single classification.

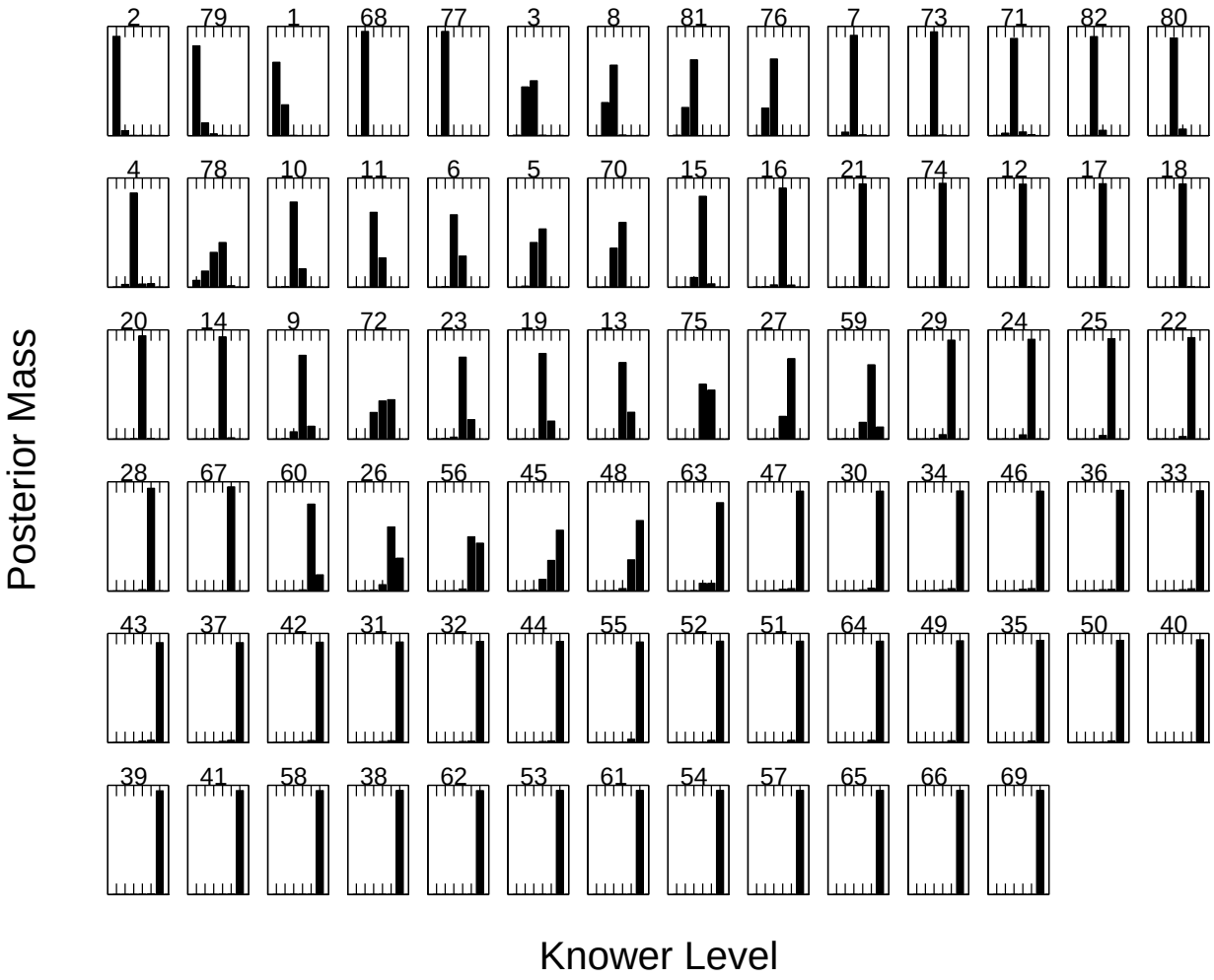


Figure 12.3: Posterior over knower levels for all 82 subjects.

When inferring a discrete latent variable representing a class, highly-peaked posteriors like this are a good sign that the model is a useful one. When models are badly mis-specified, Bayesian inference naturally does 'model averaging' blending over a mixture of possibilities to try and fit the data, but making interpretation difficult.

Figure 12.4 shows the relationship between the posterior predictions of the model and the observed behaviors for six children, chosen to span a range of knower levels and be among those children who answered the most questions. Each panel correspond to a child, with the x-axis giving the question asked and the y-axis giving the answer given. The size of the squares in each cell correspond to the posterior probability that the child will give that many toys when asked that question. The overlayed crosses are the raw behavioral data.

Figure 12.5 shows the same sort of analysis for the posterior predictive for each knower level. The overlayed data now include every child inferred (via their MAP estimate[1] and this refers to a point estimate of the posterior distribution, namely the mode or highest point) to have that knower level.

[1]MAP stands for "maximum a posteriori"

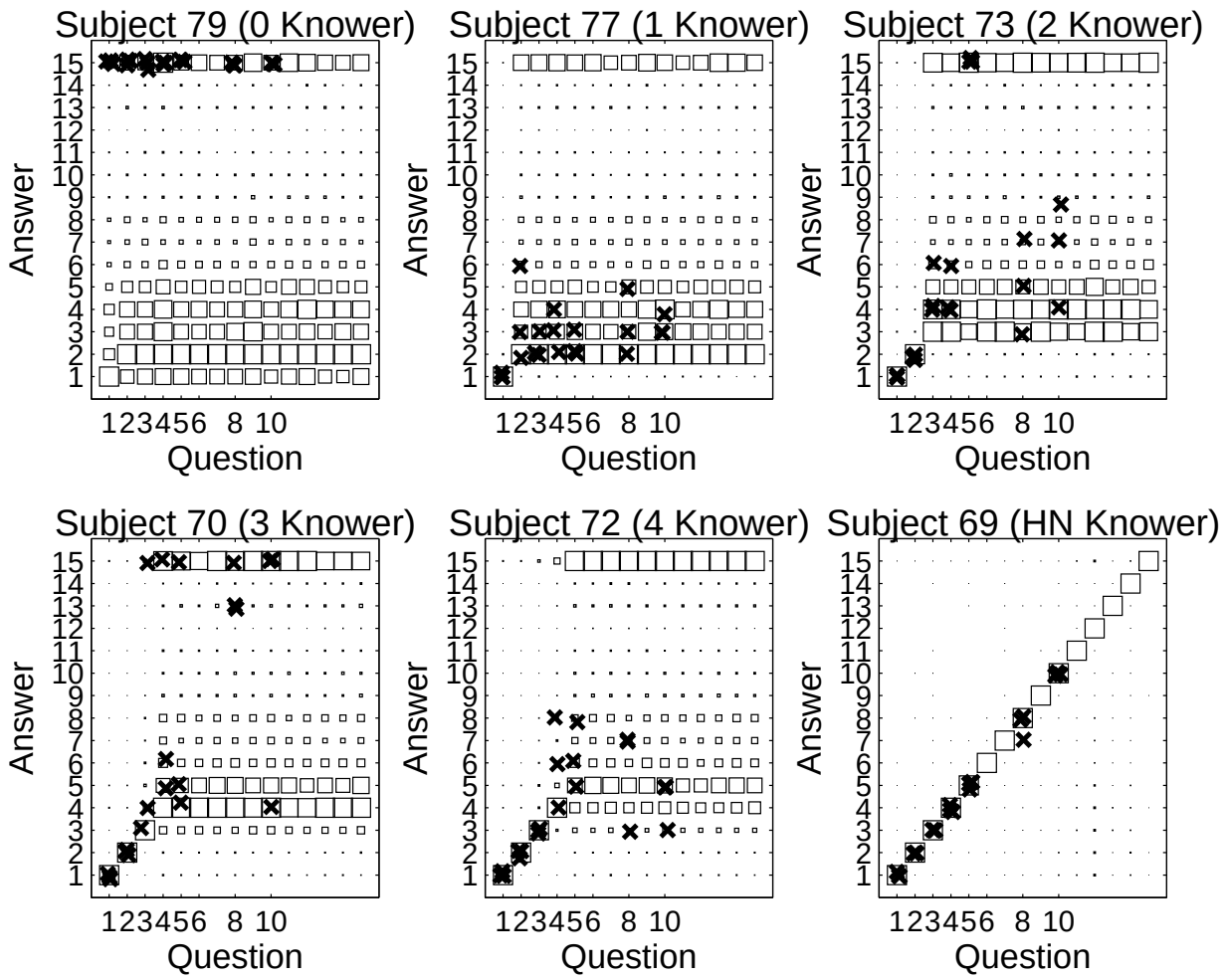


Figure 12.4: Posterior prediction for each of the knower levels, for each possible question and answer combination, shown by squares with sizes proportion to the posterior mass. Overlayed are the behavioral data according their inferred most likely knower level.

Exercises

1. Interpret the distinctive visual patterns of posterior prediction in Figure 12.5. If you can do this, I'd say you understand the model, and knower level theory.
2. Think hard about the predictions of the knower level theory in the case of 4-knowers, and how this generates the posterior predictions in Figure 12.5. You should see there are a few observed data that violate the predictions. Assume you want to argue for the knower level theory. How can you explain these data? (Hint: Look at the data for subject 72 in Figure 12.4, and their posterior over knower levels Figure 12.3). If you can work all this through, I'd say you get Bayesian inference.
3. What do you think of the MAP estimate summary of the posterior uncertainty about knower levels? What might be a better alternative in this case?

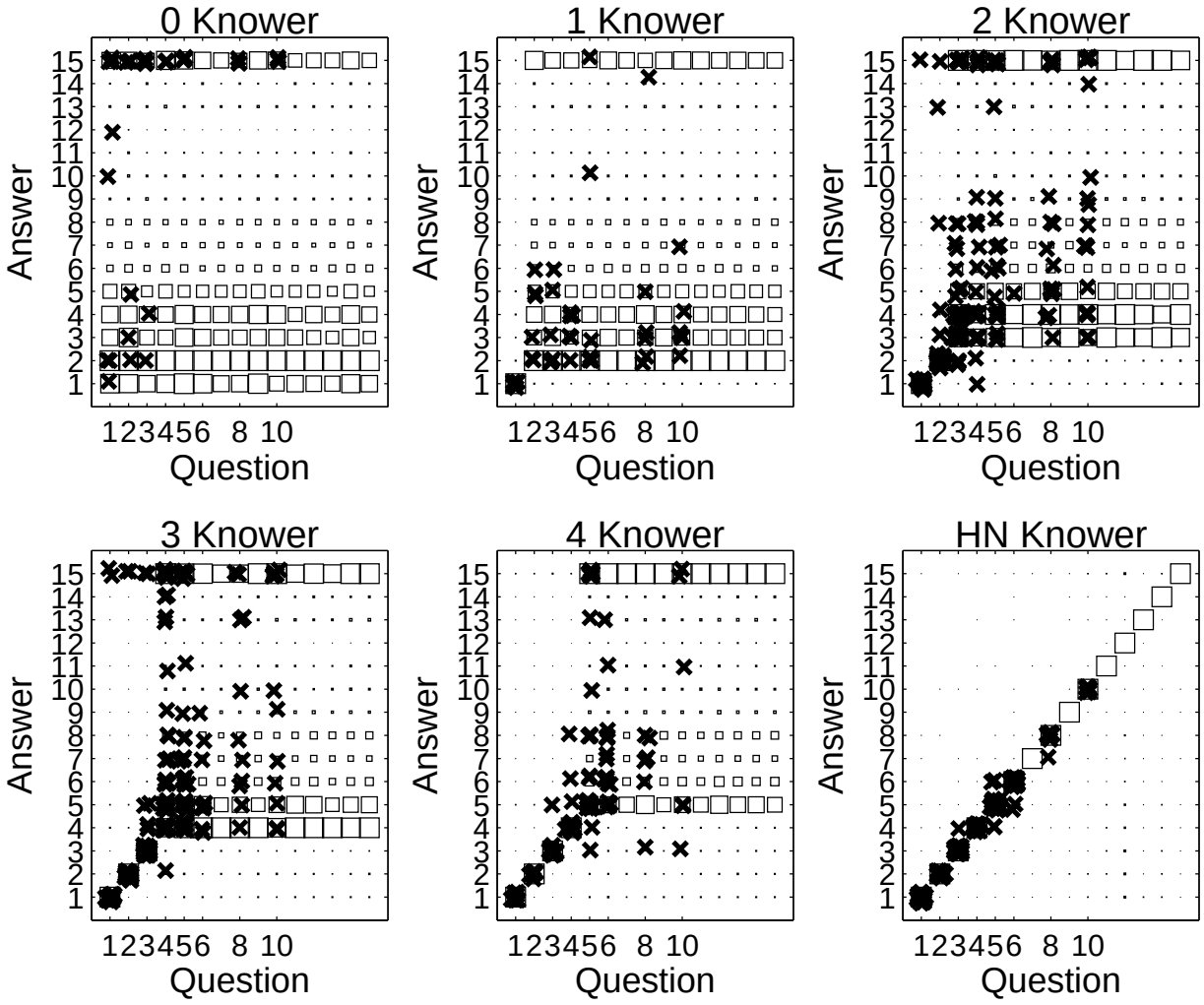
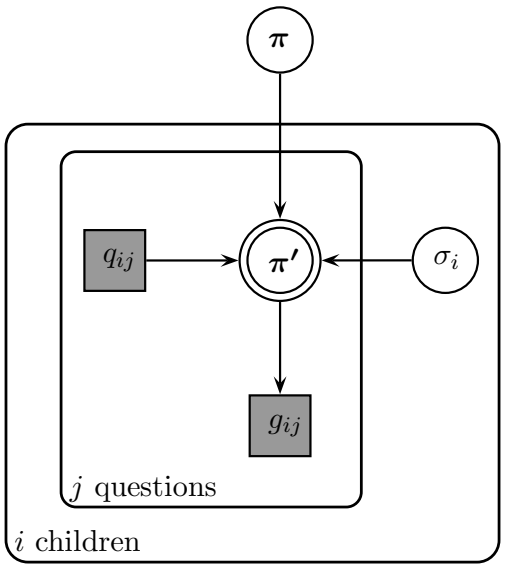


Figure 12.5: Posterior prediction for six selected subjects according to the knower level model.

12.2 Analog Representation Model



$$\boldsymbol{\pi} \sim \text{Dirichlet}(1, \ldots, 1)$$

$$\sigma_i \sim \text{InvSqrtGamma}(0.001, 0.001)$$

$$\pi'_{ijk} \propto \pi_k \times \exp\left\{-\tfrac{1}{2}\left(\tfrac{k-q_{ij}}{q_i\sigma_i}\right)^2\right\}$$

$$g_{ij} \sim \text{Categorical}(\boldsymbol{\pi'})$$

Figure 12.6: Graphical model for behavior on the give-n task according to the analog representation theory.

Figure 12.6 presents a graphical model for the give-n task using the alternative analog representation model. The same basic idea of a base-rate being updated by the questions still applies. Now, though, the updating is done according to a Gaussian function. This Gaussian in centered on the number asked for, and has a standard deviation that is proportional to the mean (i.e., bigger numbers are estimated less precisely). The constant of proportionality relating the mean and standard deviation is σ_i for the ith subject, and is called the "coefficient of variation" in the literature.

The following code implements this graphical model in WinBUGS. Notice how the Gaussian function is sampled at discrete points corresponding to the possible behaviors, and then normalized.

```
% Number Concept Using Analog Representation
model {
   # Data
   for (i in 1:S){
      for (j in 1:Q[i]){
         # Probability g[i,j] when asked for q[i,j]
         # is a categorical draw from their distribution over the 1:N toys
         # which is proportional to Gaussian centered on q[i] with standard deviation
         # given by coefficient of variation times the center
         g[i,j] ~ dcat(npiprime[i,q[i,j],1:N])
      }
   }
   # Likelihood
   for (i in 1:S){
      for (j in 1:N){
         for (k in 1:N){
            lik[i,j,k] <- 1/(sqrt(2*pic)*sigma[i]*j)*exp(-pow(k-j,2)/(2*pow(sigma[i]*j,2)))
            piprime[i,j,k] <- lik[i,j,k]*pi[j]
            npiprime[i,j,k] <- piprime[i,j,k]/sum(piprime[i,j,1:N])
```

```
      }
    }
  }
  # Posterior Prediction
  for (i in 1:S){
    for (j in 1:N){
      for (k in 1:N){
        npiprime.cut[i,j,k] <- cut(npiprime[i,j,k])
      }
      pp[i,j] ~ dcat(npiprime.cut[i,j,1:N])
    }
  }
  # Base Rate
  for (i in 1:N){
    pitmp[i] ~ dunif(0,1)
    pi[i] <- pitmp[i]/sum(pitmp[1:N])
    pi.cut[i] <- cut(pi[i])
  }
  ppb ~ dcat(pi.cut[1:N])
  # Scalar Variability Coefficient
  for (i in 1:S){
    lambda[i] <- 1/pow(sigma[i],2)
    sigma[i] ~ dunif(0,10)
  }
  # Constants
  pic <- 3.1415
}
```

The code `NumberConcept_2.m` (Matlab) or `NumberConcept_2.R` (R) applies the model to the same behavioral data as before, and produces some analyses. Figure 12.7 shows the base-rate that is inferred, which is much less convincing than was found using the knower level theory model.

Finally, Figure 12.8 shows the relationship between the posterior predictions of the model and the observed behaviors for the same six children considered in Figure 12.5. These children also span a range of coefficients of variation, and we note that many children have estimates much larger than the 0.2-ish range espoused by proponents of the theory (who typically do not consider the give-n task).

Exercises

1. Interpret the distinctive various patterns of posterior prediction in Figure 12.8. If you can do this, I'd say you understand the model, and analog representation theory.
2. Why, if we expect $\sigma \approx 0.2$ are there large coefficients of variation?
3. What would you say about the prediction for subject 79?

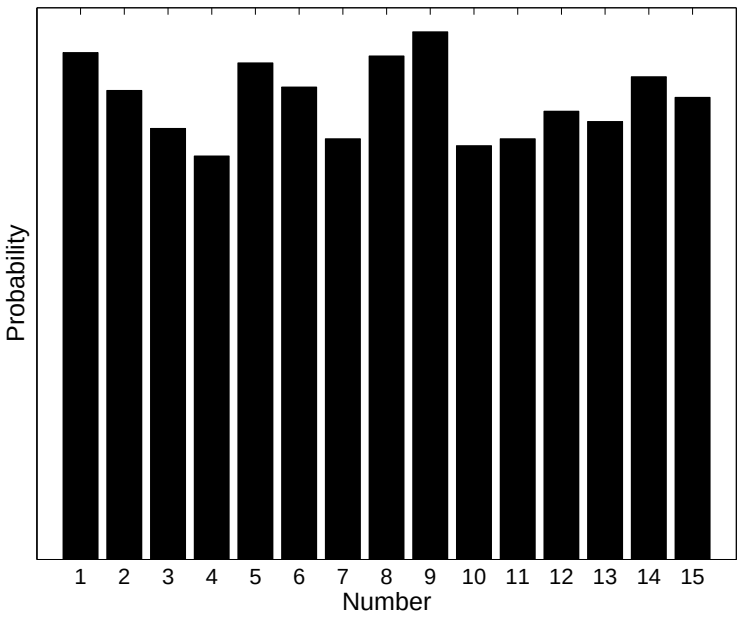


Figure 12.7: Posterior base rate for giving $1, \ldots, 15$ toys using the analog representation model.

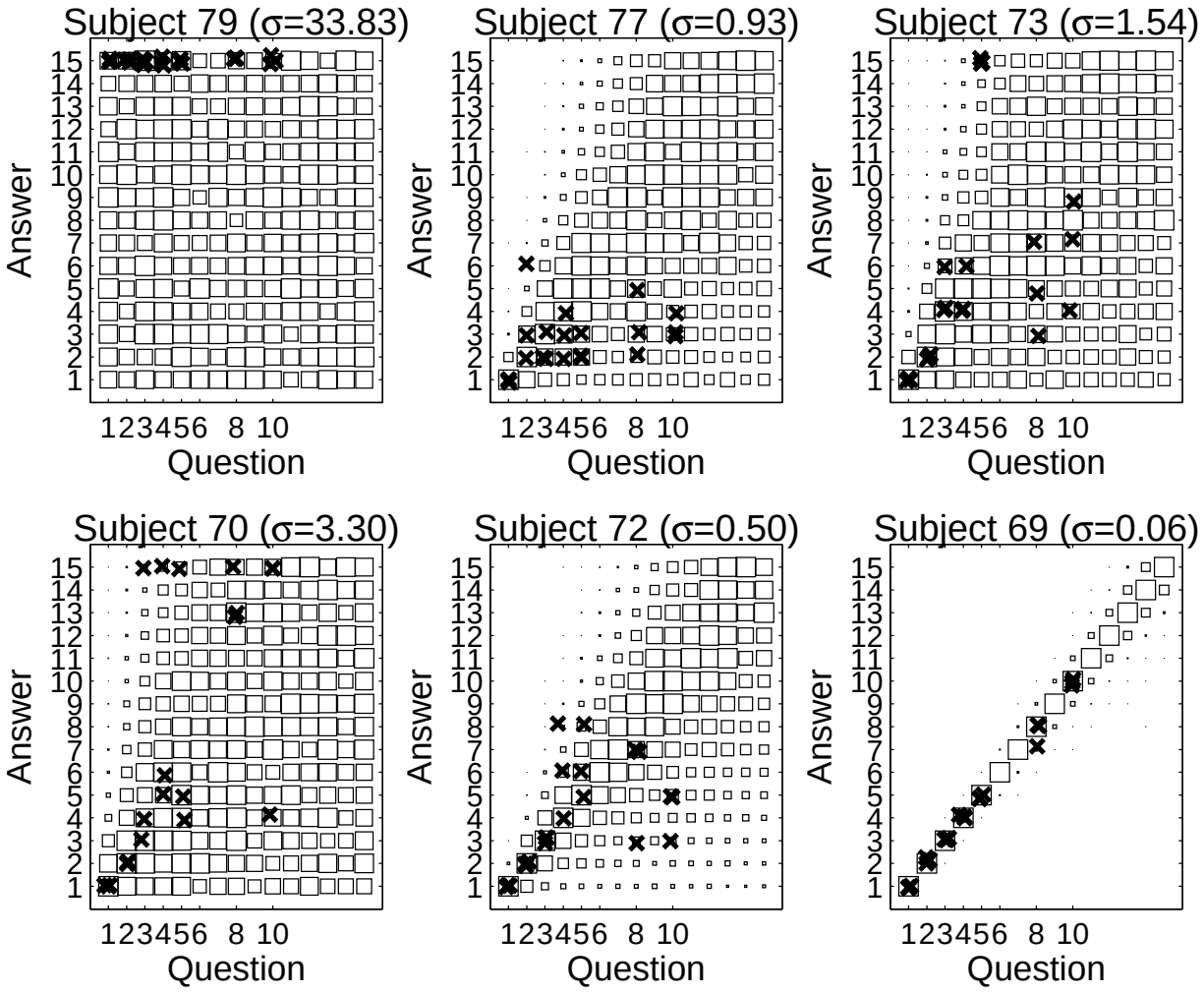


Figure 12.8: Posterior prediction for six selected subjects according to the analog representation model.

CHAPTER 13

THE SIMPLE MODEL OF MEMORY

Brown, Neath, and Chater (2007) proposed a temporal ratio model of memory called SIMPLE. The model assumes memories are encoded with a temporal component, but that the representations are logarithmically compressed, so that more distant memories are more similar. The model also assumes distinctiveness play a central role in performance on memory tasks, and that interference rather than decay is responsible for forgetting. And, perhaps most importantly, SIMPLE assumes the same memory processes operate at all time scales, unlike theories and models that assume different short- and long-term memory mechanisms.

We will focus on the first application considered by Brown et al. (2007), which involves seminal immediate free recall data reported by Murdock (1962). The data give the proportion of words correctly recalled averaged across participants, for lists of 10, 15, and 20 words presented at a rate of 1 s per word, and lists of 20, 30 and 40 words presented at a rate of 2 s per word.

Brown et al. (2007) make some reasonable assumptions about undocumented aspects of the task (e.g., the mean time of recall from the end of list presentation), to set the time, T_i between the learning and retrieval of the ith item. With these times established, the application of the SIMPLE the free recall data involves five stages, as conveniently described in Brown et al. (2007, Appendix).

First, the ith presented item, associated with time T_i, is represented in memory using logarithmic compression, given by $M_i = \log T_i$. Secondly, the similarity between each pair of items is calculated as $\eta_{ij} = \exp(-c|M_i - M_j|)$, where c is a parameter measuring the "distinctiveness" of memory. Thirdly, the discriminability of each pair of items is calculated as $d_{ij} = \eta_{ij} / \sum_k \eta_{ik}$. Fourthly, the retrieval probability of each pair of items is calculated as $r_{ij} = 1/(1 + \exp(-s(d_{ij} - t)))$, where t is a threshold parameter and s is a threshold noise parameter. Finally the probability the ith item in the presented sequence will be recalled is calculated as $\theta_i = \min(1, \sum_k r_{ik})$.

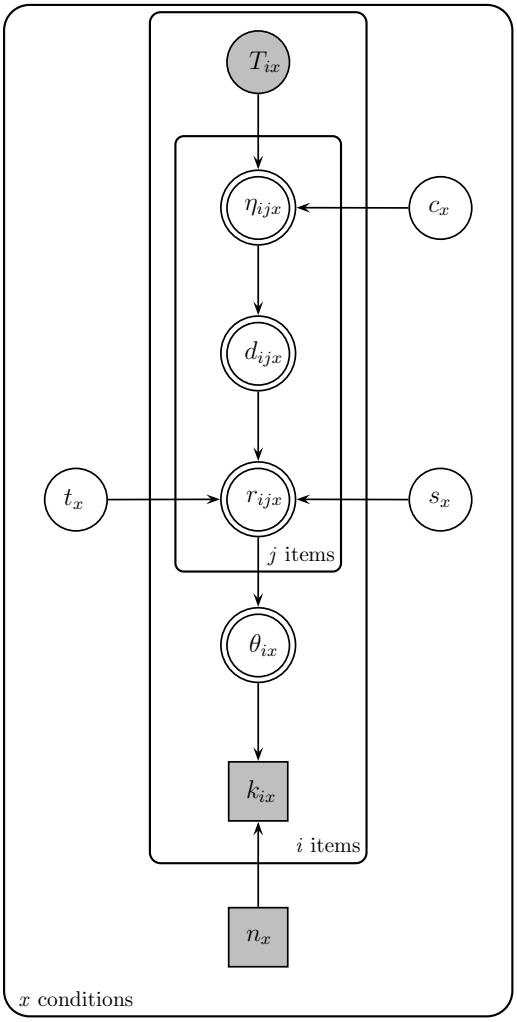
13.1 Standard SIMPLE

These stages are implemented by the graphical model shown in Figure 13.1. The graphical model has nodes corresponding to the observed times between learning and retrieval, T_i, and the observed number of correct responses k_i for the ith item and total trials n. The similarity (η_{ij}), discriminability (d_{ij}), retrieval (r_{ij}) and free recall probability (θ_i) nodes are deterministic, and simply link the times properties of the items to their accuracy of recall according to the SIMPLE model and its three parameters.

In Figure 13.1 the times, responses and free recall probabilities apply per item, and so are enclosed in a plate replicating over items. The similarity, discriminability and retrieval measures apply to pairs of variables, and so involve an additional plate also replicating over items. We follow Brown et al. (2007) by fitting the c, t, and s parameters independently for each condition. This means the entire graphical model is also enclosed in a plate replicating over the $x = 1, \ldots, 6$ conditions in the Murdock (1962) data.

The WinBUGS code to implement the graphical model is fairly straightforward. Notice that the posterior predictive is calculated in some detail, leading up to `pcpred`, which is the posterior predicted accuracy.

```
# SIMPLE Model
model{
   # Likelihood
   for (d in 1:dsets){
      for (i in 1:listlength[d]){
         k[i,d] ~ dbin(theta[i,d],tr[d])
      }
   }
   # Similarities, Discriminabilities, and Response Probabilities
   for (d in 1:dsets){
```



$c_x \sim \text{Uniform}(0, 100)$

$s_x \sim \text{Uniform}(0, 100)$

$t_x \sim \text{Uniform}(0, 1)$

$\eta_{ijx} = \exp\left(-c_x \left|\log T_{ix} - \log T_{jx}\right|\right)$

$d_{ijx} = \frac{\eta_{ijx}}{\sum_k \eta_{ikx}}$

$r_{ijx} = \frac{1}{1+\exp(-s_x(d_{ijx}-t_x))}$

$\theta_{ix} = \min(1, \sum_k r_{ikx})$

$k_{ix} \sim \text{Binomial}(\theta_{ix}, n_x)$

Figure 13.1: Graphical model implementing the SIMPLE model of memory.

```
   for (i in 1:listlength[d]){
      for (j in 1:listlength[d]){
         # Similarities
         S[i,j,d] <- exp(-c[d]*abs(log(T[i,d])-log(T[j,d])))
         # Discriminabilities
         D[i,j,d] <- S[i,j,d]/sum(S[i,1:listlength[d],d])
         # Response Probabilities
         R[i,j,d] <- 1/(1+exp(-s[d]*(D[i,j,d]-t[d])));
      }
      # Free Recall Overall Response Probability
      theta[i,d] <- min(1,sum(R[i,1:listlength[d],d]))
   }
}
# Priors And Cuts For Posterior Predictive
for (d in 1:dsets){
   c[d] ~ dunif(0,100)
   s[d] ~ dunif(0,100)
```

```
      t[d] ~ dbeta(1,1)
      c.cut[d] <- cut(c[d])
      s.cut[d] <- cut(s[d])
      t.cut[d] <- cut(t[d])
   }
   for (d in 1:dsets){
      for (i in 1:listlength[d]){
         kpred[i,d] ~ dbin(thetapred[i,d],tr[d])
         pcpred[i,d] <- kpred[i,d]/tr[d]
      }
   }
   for (d in 1:dsets){
      for (i in 1:listlength[d]){
         for (j in 1:listlength[d]){
            Spred[i,j,d] <- exp(-c.cut[d]*abs(log(T[i,d])-log(T[j,d])))
            Dpred[i,j,d] <- Spred[i,j,d]/sum(Spred[i,1:listlength[d],d])
            Rpred[i,j,d] <- 1/(1+exp(-s.cut[d]*(Dpred[i,j,d]-t.cut[d])));
         }
         thetapred[i,d] <- min(1,sum(Rpred[i,1:listlength[d],d]))
      }
   }
}
```

The code `Simple_1.m` (Matlab) or `Simple_1.R` (R) involves some initial steps to get the data organized before passing it to WinBUGS, and analysis in the form of graphs of the posterior predictions and joint posterior space. Both of these graphs (and especially the joint posterior) have been tuned a bit for this specific analysis, and so would need some more effort to remove the 'magic numbers' and be applied robustly to other cases. But it might be worth the effort, because the joint posterior plot has a lot of promise, I think.

The posterior predictive analysis should look something like Figure 13.2, which shows the posterior prediction for the six Murdock (1962) data sets. The solid lines show the probability the item in each serial position was correctly recalled. A total of 20 samples from the posterior predictive are shown for each serial position as gray points, making a gray area that spans the range in which the model expects the data to lie.

The posterior analysis should look something like Figure 13.3, which shows the joint posterior parameter distribution as a three-dimensional plot, with 20 posterior samples for each condition shown by different markers. Also shown, projected onto the planes are the pairwise joint distributions of each possible combination of parameters (marginalized over the other parameter in each case). Finally, the marginal distributions for each parameter are shown along the three axes.

Figure 13.3 attempts to convey the detailed information about the distinctiveness, threshold, and threshold noise parameters provided by the computational Bayesian approach. The point estimates of the original analysis are now extended to include information about variability and co-variation. This additional information is important to understanding how parameters should be interpreted, and for suggesting model development. For example, the lack of overlap of the three-dimensional points for the six conditions suggests that there are important differences in model parameters for different item list lengths and presentation rates. In particular, it seems unlikely that an alternative approach to fitting the six conditions using a single discriminability level and threshold function will be adequate.

Another intuition, this time coming from the two-dimensional joint posteriors is that there is a trade-off between the threshold and threshold noise parameters, since their joint distributions (shown by the points in the bottom plane) show a high level of correlation for all of the conditions. This means that the data in each condition are consistent with relatively high thresholds and relatively low levels of threshold noise, or with relatively low thresholds and relatively high levels of threshold noise. This is probably not an ideal state of affairs: generally parameters are more easily interpreted and theoretically compelling if they operate independently of each other. In this way, the information in the joint parameter posterior suggests an area in which the model might need further development or refinement.

As a final example of the information in the joint posterior, we note that the marginal distributions for

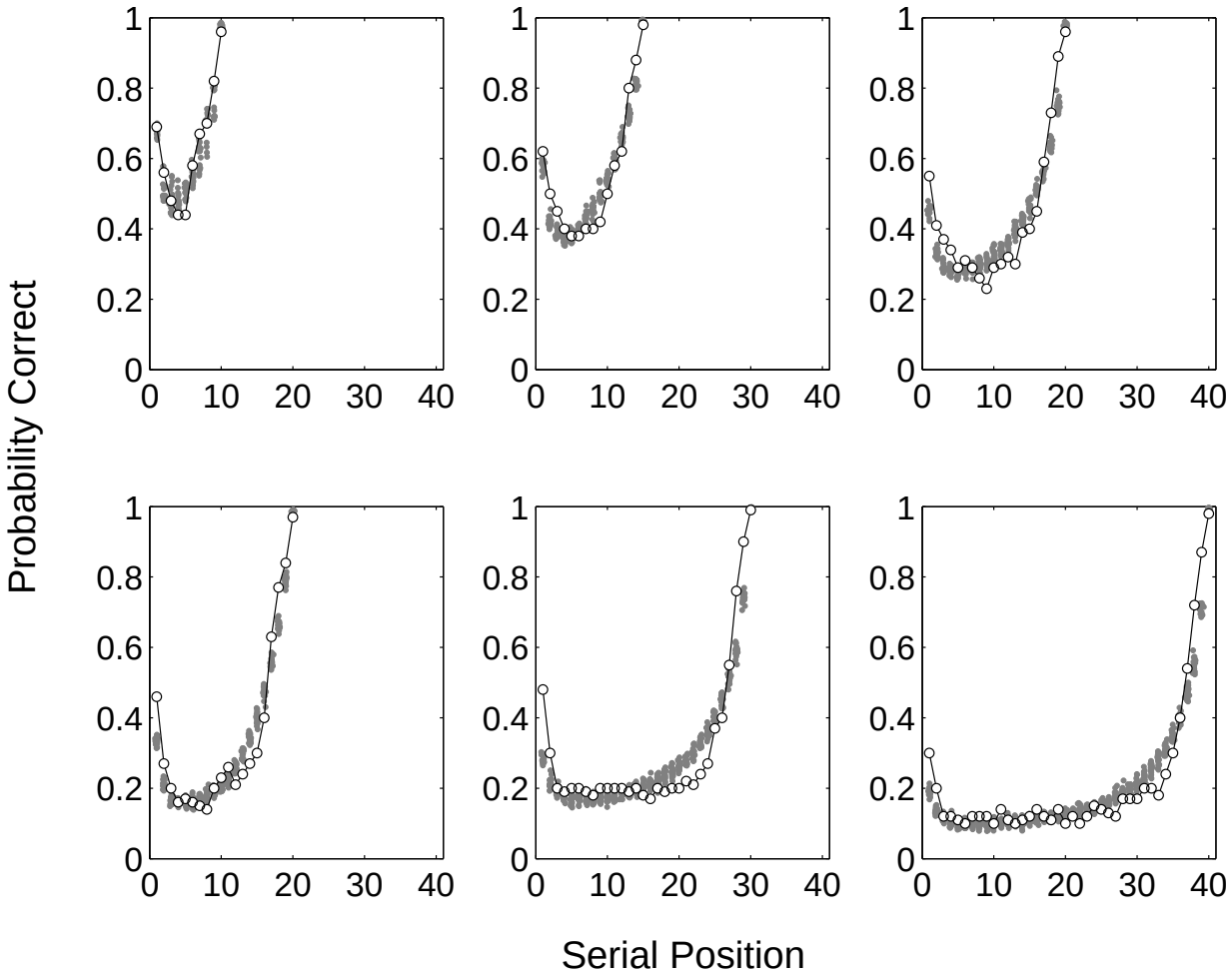


Figure 13.2: Posterior prediction of SIMPLE model for the six conditions of the Murdock (1962) immediate free recall data. The solid lines show the data, and the gray areas show 20 posterior predictive samples for the item at each serial position. The conditions are labeled according to the number of items and the rate of presentation, so that, for example, the '10-2' condition had ten items presented at 1 s per item.

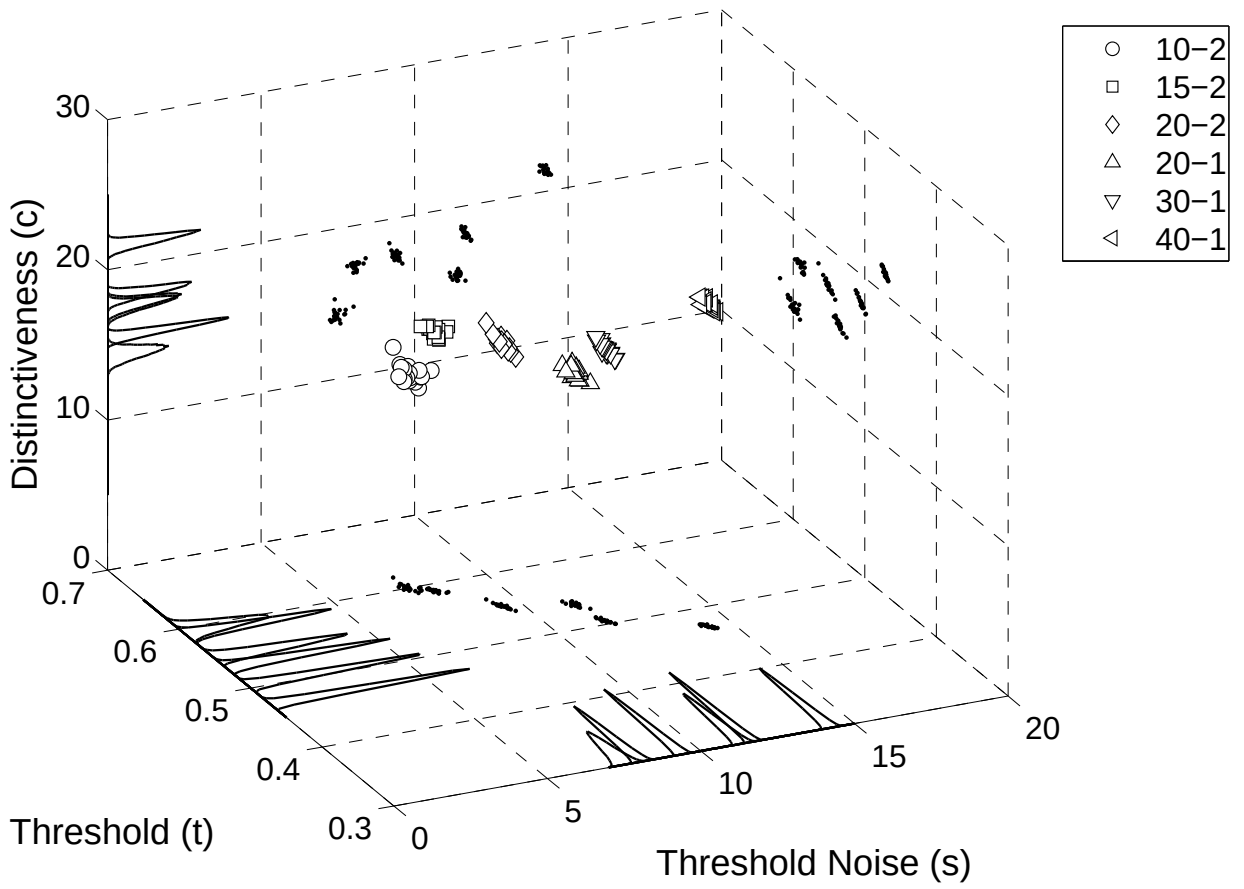


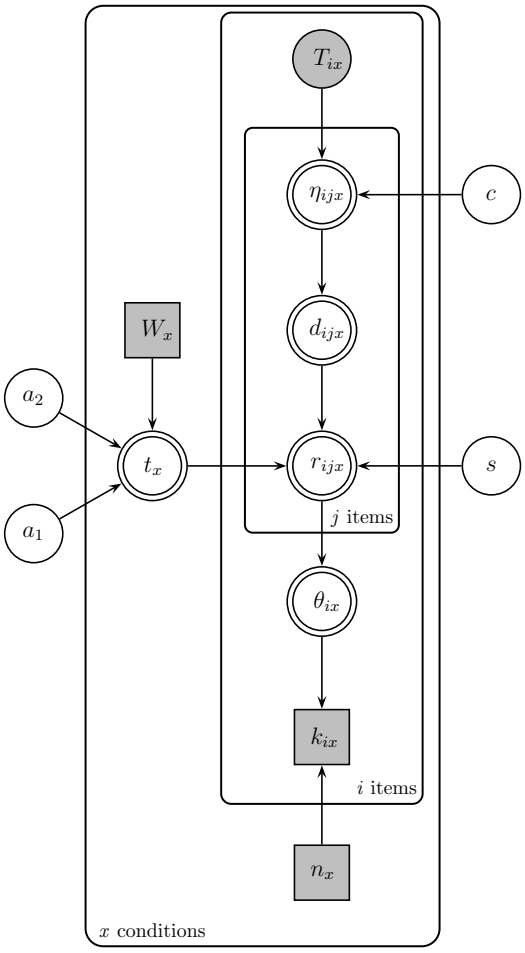
Figure 13.3: Joint posterior parameter space for the SIMPLE model for the six conditions of the Murdock (1962) immediate free recall data.

the threshold parameter shown in Figure 13.3 seem to show a systematic relationship with item list length. In particular, the threshold decreases as the item list length increases from 10 to 40, with overlap between the two conditions with the most similar lengths (i.e., the '10-2' and '15-2' conditions, and the '20-2' and '20-1' conditions). This type of systematic relationship suggests that, rather than treating the threshold as a free parameter, it can be modeled in terms of the known item list length. We now consider how this idea can be implemented in a hierarchical extension to the SIMPLE model.

Exercises

Here is an extended but worthwhile exercise

1. Modify the graphical model so that the same parameter values are used to account for all of the data sets. One could argue that, without some account of how c, t and s depend on properties of the task (or some more general account of how they change), it is not very useful to allow them to change freely as we and Brown et al. (2007) have done. You will also need to modify the Matlab/R code that produces the graphs.



$c \sim \text{Uniform}(0, 100)$

$s \sim \text{Uniform}(0, 100)$

$a_1 \sim \text{Uniform}(-1, 0)$

$a_2 \sim \text{Uniform}(0, 1)$

$t_x = a_1 W_x + a_2$

$\eta_{ijx} = \exp\left(-c\left|\log T_{ix} - \log T_{jx}\right|\right)$

$d_{ijx} = \frac{\eta_{ijx}}{\sum_k \eta_{ikx}}$

$r_{ijx} = \frac{1}{1+\exp(-s(d_{ijx}-t_x))}$

$\theta_{ix} = \min(1, \sum_k r_{ikx})$

$k_{ix} \sim \text{Binomial}(\theta_{ix}, n_x)$

Figure 13.4: Graphical model implementing a hierarchical extension to the SIMPLE model of memory.

13.2 A Hierarchical Extension of SIMPLE

Our hierarchical Bayesian extension of SIMPLE is represented by the graphical model shown in Figure 13.4. There are two important changes from the model that replicated the assumptions of Brown et al. (2007). First, the distinctiveness (c) and threshold noise (s) parameters are now assumed to have the same value for all experimental conditions. In Figure 13.4, their nodes are outside the plate replicated over conditions, and they are no longer indexed by x. We do not believe this is a theoretically realistic assumption (indeed, as we pointed out, the joint posterior in Figure 13.3 argues against it), but it makes for a good tutorial example.

It is the second change that captures this main point, and corresponds to the way the thresholds t_x are determined. Rather than being assumed to be independent, these threshold now depend on the item list length, denoted W_x for the xth condition, via a linear regression function $t_x = a_1 W_x + a_2$ parameterized by the coefficients a_1 and a_2. Consistent with the intuitions gained from Figure 13.3, we make the assumption the linear relationship expresses a decrease in threshold as item list length increases, by using the prior $a_1 \sim \text{Uniform}\big(-1, 0\big)$.

The goal of our hierarchical extensions is to move away from thinking of parameters as psychological variables that vary independently for every possible immediate serial recall task. Rather, we now conceive of the parameters as psychological variables that themselves now need explanation, and attempt to model how they change in terms of more general parameters.

This approach not only forces theorizing and modeling to tackle new basic questions about how serial recall processes work, but also facilitates evaluation of the prediction and generalization capabilities of the basic model. By making the threshold parameter depend on characteristics of the task (i.e., the number of words in the list) in a systematic ways, and by treating the other parameters as invariant, our hierarchical extension automatically allows SIMPLE to make predictions about other tasks.

The WinBUGS code to implement the hierarchical graphical model follows.

```
# Hierarchical SIMPLE Model
model{
   # Likelihood
   for (d in 1:dsets){
      for (i in 1:listlength[d]){
         k[i,d] ~ dbin(theta[i,d],tr[d])
      }
   }
   # Similarities, Discriminabilities, and Response Probabilities
   for (d in 1:dsets){
      t[d] <- max(0,min(1,alpha[1]*W[d]+alpha[2]))
      for (i in 1:listlength[d]){
         for (j in 1:listlength[d]){
            # Similarities
            S[i,j,d] <- exp(-c*abs(log(T[i,d])-log(T[j,d])))
            # Discriminabilities
            D[i,j,d] <- S[i,j,d]/sum(S[i,1:listlength[d],d])
            # Response Probabilities
            R[i,j,d] <- 1/(1+exp(-s*(D[i,j,d]-t[d])));
         }
         # Free Recall Overall Response Probability
         theta[i,d] <- min(1,sum(R[i,1:listlength[d],d]))
      }
   }
   # Priors
   c ~ dunif(0,100)
   s ~ dunif(0,100)
   alpha[1] ~ dunif(-1,0)
   alpha[2] ~ dunif(0,1)
   # Posterior Predictive
   c.cut <- cut(c)
   s.cut <- cut(s)
   for (i in 1:2){
      alpha.cut[i] <- cut(alpha[i])
   }
   for (d in 1:gsets){
      for (i in 1:listlength[d]){
         kpred[i,d] ~ dbin(thetapred[i,d],tr[d])
         pcpred[i,d] <- kpred[i,d]/tr[d]
      }
   }
   for (d in 1:gsets){
      t.cut[d] <- max(0,min(1,alpha.cut[1]*W[d]+alpha[2]))
      for (i in 1:listlength[d]){
         for (j in 1:listlength[d]){
            Spred[i,j,d] <- exp(-c.cut*abs(log(T[i,d])-log(T[j,d])))
            Dpred[i,j,d] <- Spred[i,j,d]/sum(Spred[i,1:listlength[d],d])
```

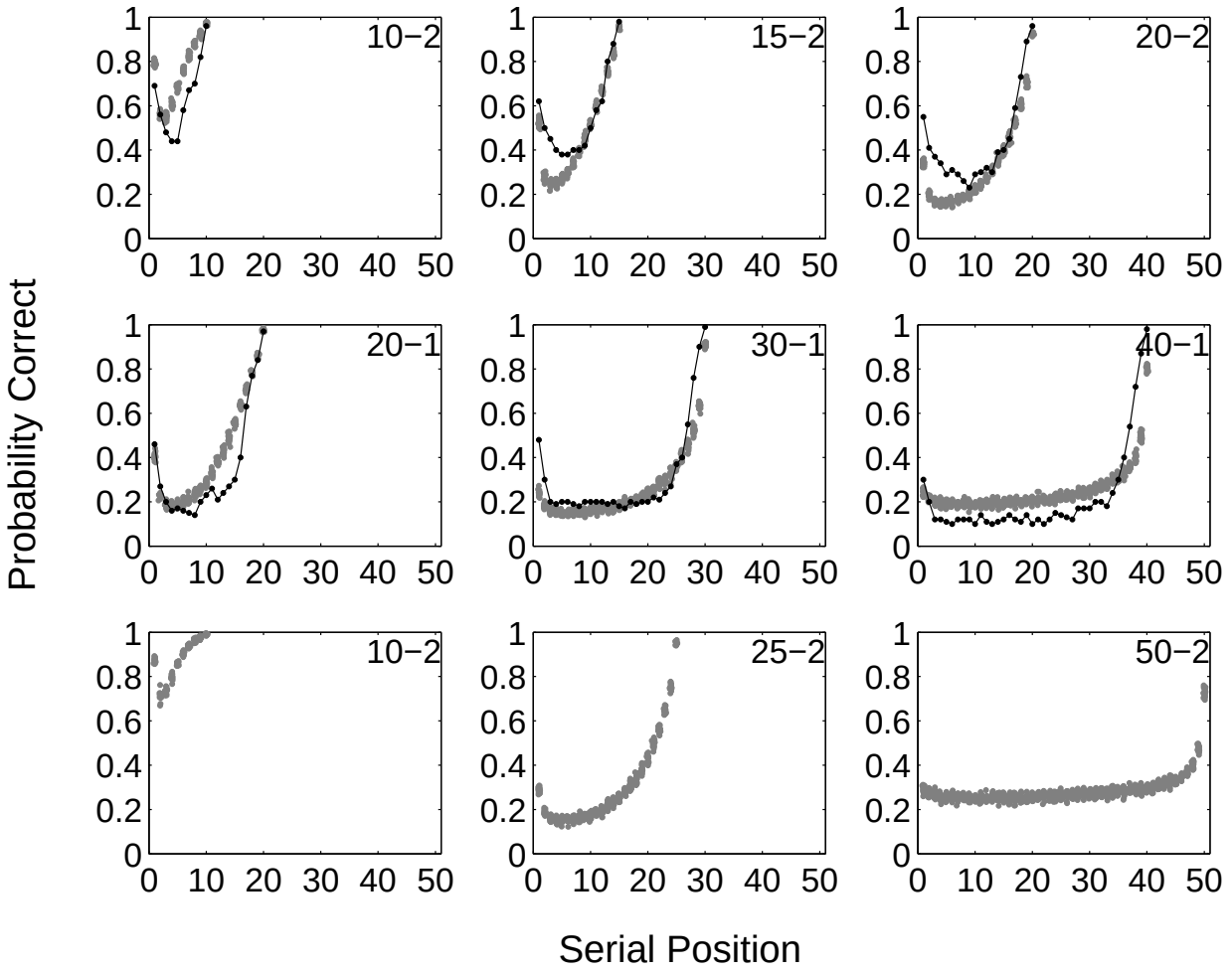


Figure 13.5: Posterior prediction of the hierarchical extension of the SIMPLE model for the six conditions of the Murdock (1962) immediate free recall data, and in generalizing to three new conditions. The solid lines show the data, and the gray areas show 20 posterior predictive samples for the item at each serial position.

```
            Rpred[i,j,d] <- 1/(1+exp(-s.cut*(Dpred[i,j,d]-t.cut[d])));
          }
          thetapred[i,d] <- min(1,sum(Rpred[i,1:listlength[d],d]))
        }
    }
}
```

The code `Simple_2.m` (Matlab) or `Simple_2.R` (R) applies the hierarchical model to the Murdock (1962) conditions, and also to three other possible conditions, for which data are not available. These generalization conditions all involve presentations rates of 1 s per item, but with 10, 25, and 50 items, corresponding to both interpolations and extrapolations relative to the collected data.

The posterior predictive performance is shown in Figure 13.5. The top two rows show the Murdock (1962) conditions, while the bottom row shows the predictions the model makes about the generalization conditions.

The posterior analysis should look something like Figure 13.6, which shows the modeling inferences about

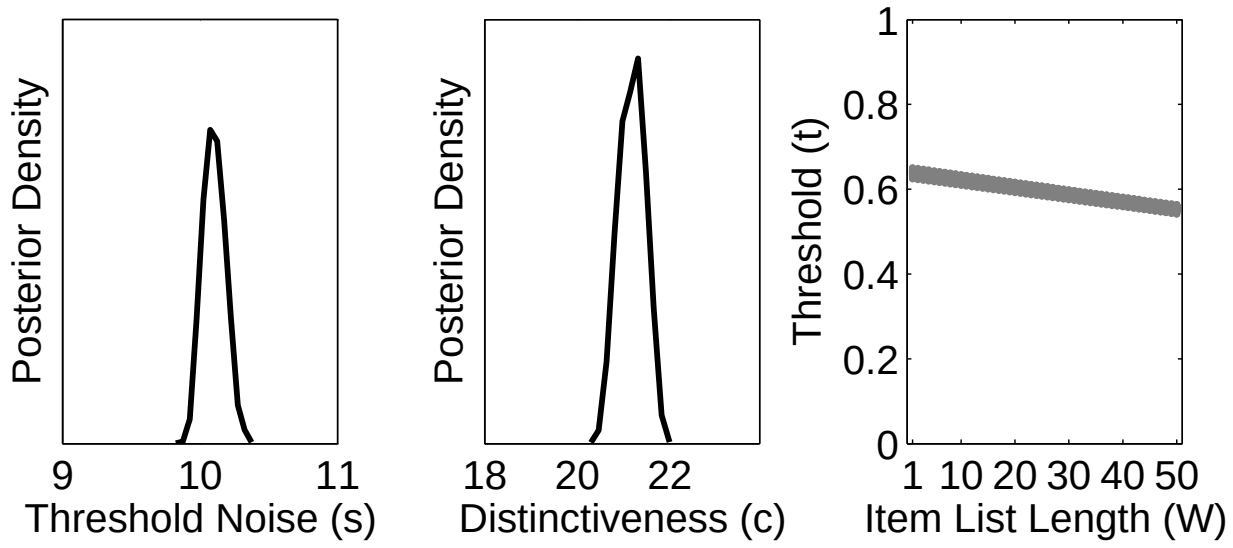


Figure 13.6: Posteriors for the SIMPLE model parameters in its hierarchically extended form.

the distinctiveness, threshold noise and threshold parameters. For the first two of these, the inferences take the form of single posterior distributions. For the threshold parameter, however, the posterior inference is now about its functional relationship to item list length. The posterior distribution for this function is represented in the right panel of Figure 13.6 by showing 50 posterior samples at each possible length $W = 1, \ldots, 50$. These posterior samples are found by taking joint posterior samples (a_1, a_2) and finding $t = a_1 W + a_2$ for all values of W.

Exercises

Here is another challenging but worthwhile exercise

1. Look back at the results for the original, non-hierarchical, analysis of SIMPLE. Find another avenue towards doing a hierarchical extension, and implement it.

Part III

Hypothesis Testing for Statistical Models

CHAPTER 14

BAYESIAN HYPOTHESIS TESTING

Up to this point we have concerned ourselves with parameter estimation, implicitly taking the appropriateness of the underlying model for granted. In much of social science, however, researchers entertain more than just a single statistical model. In fact, the statistical models often represent competing theories or hypotheses, and the focus of interest is on which substantive theory or hypothesis is more correct, more plausible, and better supported by the data. For example, researchers might want to know whether the improvement of performance with practice follows a power function or an exponential function. As another example, we might want to know the extent to which your performance in our test (i.e., 9 correct answers out of 10 questions) is consistent with the hypothesis that you were just guessing. This may involve a test of $M_1 : \theta = 0.5$ versus $M_2 : \theta \neq 0.5$.

The fundamental and general Bayesian solution to the foregoing model selection of hypothesis testing problems is as follows. For simplicity, assume that you contemplate two alternative accounts of the data, M_1 and M_2, and that you seek to quantify model uncertainty in terms of probability. Consider first M_1. Bayes' rule dictates how your prior probability of M_1, $p(M_1)$, is updated through the observed data D to give the posterior probability of M_1, $p(M_1|D)$:

$$p(M_1|D) = \frac{p(M_1)p(D|M_1)}{p(M_1)p(D|M_1) + p(M_2)p(D|M_2)}. \tag{14.1}$$

In the same way, one can calculate the posterior probability of M_2, $p(M_2|D)$. The ratio of these posterior probabilities is given by

$$\frac{p(M_1|D)}{p(M_2|D)} = \frac{p(M_1)}{p(M_2)} \frac{p(D|M_1)}{p(D|M_2)}. \tag{14.2}$$

This equation shows that the change from prior odds $p(M_1)/p(M_2)$ to posterior odds $p(M_1|D)/p(M_2|D)$ is determined entirely by the ratio of the marginal likelihoods $p(D|M_1)/p(D|M_2)$. This ratio is generally known as the *Bayes factor* (Jeffreys, 1961), and the Bayes factor, or the log of it, is often interpreted as the *weight of evidence* coming from the data (Good, 1985). A hypothesis test based on the Bayes factor prefers the model under which the observed data are most likely (for details see Berger & Pericchi, 1996; Bernardo & Smith, 1994, Chapter 6; Gill, 2002, Chapter 7; Klugkist, Laudy, & Hoijtink, 2005; Kass & Raftery, 1995; MacKay, 2003; Myung & Pitt, 1997; O'Hagan, 1995).

Thus, when the Bayes factor for M_1 versus M_2 equals 2 (i.e., $BF_{12} = 2$), this means that the data are twice as likely to have occurred under M_1 than under model M_2. When the prior odds are equal, such that M_1 and M_2 are equally likely a priori, the Bayes factors can be converted to posterior probabilities: $p(M_1|D) = BF_{12}/(BF_{12} + 1)$. This means that $BF_{12} = 2$ translates to $p(M_1|D) = 2/3$.

To illustrate, consider again our binomial example of 9 correct responses out of 10 questions, and the test between two models for performance: guessing (i.e., $M_1 : \theta = 0.5$) versus not guessing (i.e., $M_2 : \theta \neq 0.5$). From Equation 1.1, the marginal likelihood for M_1, $p(D|M_1)$, is simply $\binom{10}{9}\left(\frac{1}{2}\right)^{10}$. The marginal likelihood for model M_2 is more difficult to calculate, as θ is a free parameter. In general, the marginal likelihood is obtained by integrating out the model parameters in accordance with the law of total probability:

$$p(D|M_2) = \int p(D|\theta, M_2)p(\theta|M_2)d\theta. \tag{14.3}$$

This means that the marginal likelihood is computed by averaging the likelihood over the prior; conceptually, the likelihood is evaluated for every possible parameter value, weighted with its prior plausibility, and added to a summed total. When we again use the uniform distribution for θ as a prior, such that $p(\theta|M_2) \sim \text{Beta}(1, 1)$, then Equation 14.3 famously simplifies to $p(D|M_2) = 1/(n + 1)$. Thus, in our binomial example, $BF_{12} = \binom{10}{9}\left(\frac{1}{2}\right)^{10}(n + 1) \approx 0.107$. This means that the data are $1/0.107 \approx 9.3$ times more likely under M_2 than they are under M_1. With unit prior odds, the posterior probability for M_1 is $0.107/(0.107 + 1) \approx$

.10, which means that the complementary posterior probability for M_2 is approximately .90. These are probabilities assigned to hypotheses, and they are exactly what researchers want to know about.

Posterior model probabilities are not just necessary to quantify our degree of belief or preference for the candidate models under consideration. They are also necessary for Bayesian model averaging (e.g., Draper, 1995; Hoeting, Madigan, Raftery, & Volinsky, 1999; Madigan & Raftery, 1994). For instance, in a regression context we might have one model, M_1, that predicts a certain post-surgery survival rate by gender, age, weight, and history of smoking. A second model, M_2, includes two additional predictors, namely body-mass index and fitness. We compute posterior model probabilities and find that $p(M_1|D) = .6$ and consequently $p(M_2|D) = .4$. For a given patient, M_1 predicts a survival rate of 90%, and M_2 predicts a survival rate of 80%. What is our best prediction for our patient's survival rate? It is tempting to base our prediction solely on M_1, which is after all the preferred model. However, this would ignore the uncertainty inherent in the model selection procedure, and it would ignore the very real possibility that the best model is M_2, according to which the survival rate is 10% lower than it is for M_1. The Bayesian solution is to weight the two competing predictions with their associated posterior model probabilities, fully taking into account the uncertainty in the model selection procedure. In our example, the model-averaged prediction for survival rate would be $.6 \times 90\% + .4 \times 80\% = 86\%$.

Additional Advantages of Bayesian Hypothesis Testing

We have seen how Bayes factors and posterior model probabilities describe the relative support or preference for a set of candidate models, and how they can be used for model averaged predictions. Other advantages of Bayesian hypothesis testing include the following (Wagenmakers, Lee, Lodewyckx, & Iverson, 2008; see also S. J. Dennis, Lee, & Kinnell, 2008):

1. Coherence is guaranteed Suppose we have a set of three candidate models, M_1, M_2, and M_3. As

$$\frac{p(D|M_1)}{p(D|M_3)} = \frac{p(D|M_1)}{p(D|M_2)} \frac{p(D|M_2)}{p(D|M_3)}, \tag{14.4}$$

this means that $BF_{13} = BF_{12} \times BF_{23}$. For instance, when the data are five times as likely to occur under M_1 than under M_2, and seven time as likely under M_2 than under M_3, it follows that the data are $5 \times 7 = 35$ times as likely under M_1 than under M_3. No comparable result exists in classical statistics.

2. Parsimony is automatically rewarded The main challenge of hypothesis testing or model selection is to identify the model with the best predictive performance (e.g., Myung, Forster, & Browne, 2000; Wagenmakers & Waldorp, 2006). However, it is not immediately obvious how this should be done; complex models will generally provide a better fit to the observed data than simple models, and therefore one cannot simply prefer the model with the best "goodness-of-fit" – such a strategy would lead to massive overfitting. Intuition suggest that this tendency for overfitting should be counteracted by putting a premium on simplicity. This intuition is consistent with the law of parsimony or "Ockham's razor" which states that, when everything else is equal, simple models are to be preferred over complex models (Jaynes, 2003, Chapter 20; Myung & Pitt, 1997).

Formal model selection methods try to quantify the tradeoff between goodness-of-fit and parsimony. Many of these methods measure a model's overall performance by the sum of two components, one that measures descriptive accuracy and one that places a premium on parsimony. The latter component is also known as the Ockham factor (MacKay, 2003, Chapter 28). For many model selection methods, the crucial issue is how to determine the Ockham factor. One of the attractive features of Bayesian hypothesis testing is that it automatically determines the model with the best predictive performance—Bayesian hypothesis testing therefore incorporates what is known as an automatic Ockham's razor (Myung & Pitt, 1997).

To see why this is the case, consider that every statistical model makes *a priori* predictions. Complex models have a relatively large parameter space, and are therefore able to make many more predictions and cover many more eventualities than simple models. However, the drawback for complex models is that they need to spread out their prior probability across their entire parameter space. In the

limit, a model that predicts almost everything has to spread out its prior probability so thinly that the occurrence of any particular event will not greatly add to that model's credibility. As shown by Equation 14.3, the marginal likelihood for a model M is calculated by averaging the likelihood $p(D|\theta, M)$ over the prior $p(\theta|M)$. When the prior is very spread out, it will occupy a relatively large part of the parameter space in which the likelihood is almost zero, and this greatly decreases the average or marginal likelihood.

3. **Evidence can be obtained in favor of the null hypothesis** Bayesian hypothesis testing allows one to obtain evidence in favor of the null hypothesis. Because theories and models often predict the absence of a difference, it is vital for scientific progress to be able to quantify evidence in favor of the null hypothesis (e.g., Rouder, Speckman, Sun, Morey, & Iverson, 2009). In the field of visual word recognition, for instance, the entry-opening theory (Forster, Mohan, & Hector, 2003) predicts that masked priming is absent for items that do not have a lexical representation; Another example from that literature concerns the work by Bowers et al. (Bowers, Vigliocco, & Haan, 1998), who have argued that priming effects are equally large for words that look the same in lower and upper case (e.g., kiss/KISS) or that look different (e.g., edge/EDGE), a finding supportive of the hypothesis that priming depends on abstract letter identities.

   A final example comes from the field of recognition memory, where Dennis and Humphreys' *bind cue decide model of episodic memory* (BCDMEM) predicts the absence of a list-length effect and the absence of a list-strength effect (S. Dennis & Humphreys, 2001). This radical prediction of a null effect allows researchers to distinguish between context-noise and item-noise theories of inference in memory (S. J. Dennis et al., 2008). In Bayesian statistics, the null hypothesis has no special status, and evidence for it is quantified just as it is for any other hypothesis. In classical statistics, support for informative predictions from null hypothesis can only be indirect.

4. **Evidence may be monitored as it accumulates** Bayesian hypothesis testing allows one to monitor the evidence as the data come in (Berger & Berry, 1988). In contrast to frequentist inference, Bayesian inference does not require special corrections for "optional stopping" (Wagenmakers, 2007).

   Consider, for instance, a hypothetical experiment on the neural substrate of dissociative identity disorder. In this experiment, the researcher Lisa has decided in advance to use functional magnetic resonance imaging (fMRI) to test 30 patients and 30 normal controls. Lisa inspects the data after 15 participants in each group have been tested, and finds that the results convincingly demonstrate the pattern she hoped to find. Unfortunately for Marge, she cannot stop the experiment and claim a significant result, as she would be changing the sampling plan halfway through and be guilty of "optional stopping". She has to continue the experiment, wasting not just her time and money, but also the time and efforts of the people who undergo needless testing.

   In contrast, for Bayesian hypothesis testing there is nothing wrong with gathering more data, examining these data, and then deciding whether or not to stop collecting new data – no special corrections are needed. As stated by Edwards et al. (Edwards, Lindman, & Savage, 1963), "(...) the rules governing when data collection stops are irrelevant to data interpretation. It is entirely appropriate to collect data until a point has been proven or disproven, or until the data collector runs out of time, money, or patience." (Edwards et al., 1963, p. 193).

Challenges for Bayesian Hypothesis Testing

Bayesian hypothesis testing faces two main challenges, one conceptual and one computational. The conceptual challenge is that the Bayesian hypothesis test is acutely sensitive to the shape of the prior distributions for the model parameters (e.g., Bartlett, 1957; Liu & Aitkin, in press). This distinguishes hypothesis testing from parameter estimation, in which the data quickly overwhelm the prior; the accumulation of data forces prior opinions that are very different to converge to posterior opinions that are very similar. For parameter estimation then, the choice of a prior distribution is not really all that critical unless there are very few data.

In contrast, for Bayesian hypothesis testing the prior distributions are crucial and have a lasting impact. This occurs because the marginal likelihood is an average taken with respect to the prior. Consider for instance the prior for the mean μ of a Normal distribution with known variance. One might be tempted to

use an "uninformative" prior, one that does not express much preference for one value of μ over the other. One such vague prior could be a Normal distribution with mean zero and variance 10,000. But, from a marginal likelihood perspective, this prior is consistent with almost any value of μ. When one hedges one's bets to such an extreme degree, the Bayes factor is likely to show a preference for a simple model (e.g., one in which $\mu = 0$), even when the data appear wildly inconsistent with it.

The main problem here is not that the Bayesian hypothesis test corrects for model complexity as manifested in the prior distribution. This is the automatic Ockam's razor that is an asset, not a liability, of the Bayesian hypothesis test. Instead, the problem seems to be that researchers have only a vague idea of the vagueness of their prior knowledge, or that researchers seek to use a prior that is "objective", and uses as little prior knowledge as possible. When the vagueness of the prior is arbitrary, so are the results from the Bayesian hypothesis test. When the vagueness of the prior is purposefully large, the results from the Bayesian hypothesis test tend to indicate a preference for the simple model, regardless of the data.

In order to increase the robustness of Bayesian hypothesis testing to the vagueness of the prior, several procedures have been proposed, including the local Bayes factor (Smith & Spiegelhalter, 1980), the intrinsic Bayes factor (Berger & Mortera, 1999; Berger & Pericchi, 1996), the fractional Bayes factor (O'Hagan, 1995), and the partial Bayes factor (O'Hagan, 1995; for a summary see Gill, 2002, Chapter 7). The idea of the partial Bayes factor is to sacrifice a small part of the data to obtain a posterior that is robust to the various priors one might entertain. The Bayes factor is then calculated by integrating the likelihood over this posterior instead of over the original prior. Procedures such as these are still undergoing further development and deserve more study.

The problem of vague priors is particularly evident for parameters that can take on values across the entire real line, such the mean μ of a Normal distribution. We believe that in such cases, whenever possible, the construction of a prior should be guided by the substantive knowledge in the domain of application. As Dennis Lindley has pointed out repeatedly, μ is only a Greek letter, an abstraction that may obscure the fact that it refers to something about which we have detailed prior knowledge. When μ stands for a person's weight, few rational people would assign μ an "uninformative" Normal prior distribution with mean zero and variance 10,000. Another solution is to put priors not on the means, but on effect size (Jeffreys, 1961; Rouder et al., 2009; Wetzels, Raaijmakers, Jakab, & Wagenmakers, in press); standard priors for effect size include the Cauchy (i.e, a t-distribution with one degree of freedom) and the standard Normal.

The second challenge for Bayesian hypothesis testing is that the marginal likelihood and the Bayes factor are often quite difficult to compute. Earlier, we saw that with a uniform prior on the binomial rate parameter θ (i.e., $p(\theta|M) \sim \text{Beta}(1, 1)$), the marginal likelihood simplifies from $\int p(D|\theta, M)p(\theta|M)d\theta$ to $1/(1 + n)$. However, in all but a few simple models, such simplifications are impossible. In order to be able to compute the marginal likelihood or the Bayes factor for more complex models, a series of different computational methods has been developed. A recent summary lists as many as 15 different methods (Gamerman & Lopes, 2006, Chapter 7).

For instance, one method computes the marginal likelihood through what is called the *candidates' formula* (Besag, 1989) or the *basic marginal likelihood identity* (Chib, 1995; Chib & Jeliazkov, 2001). One simply exchanges the roles of posterior and marginal likelihood to obtain

$$p(D) = \frac{p(D|\theta)p(\theta)}{p(\theta|D)}, \tag{14.5}$$

which holds for any value of θ. When the posterior is available analytically, one only needs to plug in a single value of θ and obtain the marginal likelihood immediately. This method can however also be applied when the posterior is only available through MCMC output, either from the Gibbs sampler (Chib, 1995) or the Metropolis-Hastings algorithm (Chib & Jeliazkov, 2001).

Another method that computes the marginal likelihood is to repeatedly sample parameter values from the prior, calculate the associated likelihoods, and then take the likelihood average. When the posterior is highly peaked compared to the prior—as will happen with many data or with a medium-sized parameter space—, it becomes necessary to employ more efficient sampling methods, with a concomitant increase in computational complexity.

Finally, it is also possible to compute the Bayes factor directly, without first calculating the constituent marginal likelihoods. The basic idea is to generalize the MCMC sampling routines for parameter estimation to incorporate a "model indicator" variable. In the case of two competing models, the model indicator

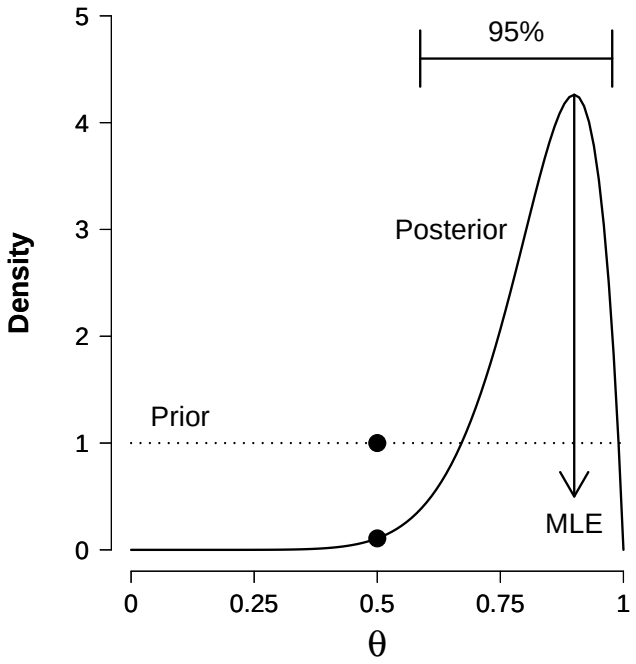


Figure 14.1: Prior and posterior distributions for binomial rate parameter θ, after observing 9 correct responses and 1 incorrect response. The mode of the posterior distribution for θ is 0.9, equal to the maximum likelihood estimate, and the 95% confidence interval extends from 0.59 to 0.98. The height of the distributions at $\theta = 0.5$ is indicated by a black dot; the ratio of these heights quantifies the evidence for $H_0 : \theta = .5$ versus $H_1 : \theta \sim \text{Beta}(1, 1)$.

variable k, say, can take on two values—for instance, $k = 1$ when the sampler is in model M_1, and $k = 2$ when the sampler is in model M_2. The Bayes factor is then estimated by the relative frequency with which $k = 1$ versus $k = 2$. This MCMC approach to model selection is called transdimensional MCMC (e.g., Sisson, 2005), an approach that encompasses both reversible jump MCMC (P. J. Green, 1995) and the product space technique (Carlin & Chib, 1995).

Almost all of these computational methods suffer from the fact that they become less efficient and more difficult to implement as the underlying models become more complex. We now turn to an alternative method, whose implementation is extremely straightforward. The methods' main limitation is that it applies only to nested models, a limitation that also holds for p-values.

14.1 The Savage-Dickey Density Ratio Test

In the simplest classical hypothesis testing framework, one contemplates two models; the null hypothesis, that fixes one of its parameters to a pre-specified value of substantive interest, say $H_0 : \phi = \phi_0$; and the alternative hypothesis, in which that parameter is free to vary, say $H_1 : \phi \neq \phi_0$. Hence, the null hypothesis is nested under the alternative hypothesis, that is, H_0 can be obtained from H_1 by setting ϕ equal to ϕ_0. Note that in the classical framework, H_0 is generally a sharp null hypothesis, or a "point null". That is, the null hypothesis states that ϕ is exactly equal to ϕ_0.

For example, in the binomial example from Figure 1.1 (shown again here as Figure 14.1) you answered 9 out of 10 questions correctly. Were you guessing or not? The classical and the Bayesian framework define $H_0 : \theta = .5$ as the null hypothesis for chance performance. The alternative hypothesis under which H_0 is nested could be defined as $H_1 : \theta \neq .5$, or, more specifically, as $H_1 : \theta \sim \text{Beta}(1, 1)$, which states that θ is free to vary from 0 to 1, and that it has a uniform prior distribution as shown in Figure 14.1.

For the binomial example, the Bayes factor for H_0 versus H_1 could be obtained by analytically integrating out the model parameter θ. However, the Bayes factor may likewise be obtained by only considering H_1, and dividing the height of the posterior for θ by the height of the prior for θ, at the point of interest. This surprising result was first published by Dickey and Lientz (1970), who attributed it to Leonard J. "Jimmie" Savage. The result is now generally known as the *Savage-Dickey density ratio* (e.g., Dickey, 1971; Gamerman & Lopes, 2006, pp. 72-74, pp. 79-80; Kass & Raftery, 1995, p. 780-781; O'Hagan & Forster, 2004, pp. 174-177; for extensions and generalizations see Chen, 2005; Verdinelli & Wasserman, 1995). Mathematically, the Savage-Dickey density ratio says that

$$BF_{01} = \frac{p(D|H_0)}{p(D|H_1)} = \frac{p(\theta = .5|D, H_1)}{p(\theta = .5|H_1)}. \tag{14.6}$$

A straightforward mathematical proof is presented in O'Hagan and Forster (2004, pp. 174-177).

In Figure 14.1, the two thick dots located at $\theta = .5$ provide the required information. It is evident from the figure that after observing 9 out of 10 correct responses, the height of the density at $\theta = .5$ has decreased, so that one would expect these data to cast doubt on the null hypothesis and support the alternative hypothesis. Specifically, the height of the prior distribution at $\theta = .5$ equals 1, and the height of the posterior distribution at $\theta = .5$ equals 0.107. From Equation 14.6 the corresponding Bayes factor is $BF_{01} = 0.107/1 = 0.107$, and this corresponds exactly to the Bayes factor that was calculated by integrating out θ.

It is clear that the same procedure can be followed when the height of the posterior is not available in closed form, but instead has to be approximated from the histogram of MCMC samples. Figure 14.2 shows the logspline estimates (Stone et al., 1997) for the prior and the posterior densities as obtained from MCMC output. The estimated height of the prior and posterior distributions at $\theta = .5$ equal 1.00 and 0.107, respectively.

In most nested model comparisons, H_0 and H_1 have several free parameters in common. These parameters are usually not of direct interest, and they are not the focus of the hypothesis test. Hence, the common parameters are known as *nuisance parameters*. For instance, one might want to test whether or not the mean of a Normal distribution is zero (i.e., $H_0 : \mu = \mu_0$ versus $H_1 : \mu \neq \mu_0$), whereas the variance σ^2 is common to both models and not of immediate interest.

In general then, the framework of nested models features a parameter vector $\theta = (\phi, \psi)$, where ϕ denotes the parameter of substantive interest that is subject to test, and ψ denotes the set of nuisance parameters. The null hypothesis H_0 posits that ϕ is constrained to some special value, i.e. $\phi = \phi_0$. The alternative hypothesis H_1 assumes that ϕ is free to vary. Now consider H_1, and let $\phi \to \phi_0$; this effectively means that H_1 reduces to H_0—it is therefore reasonable to assume that $p(\psi|\phi \to \phi_0, H_1) = p(\psi|H_0)$. In other words, when $\phi \to \phi_0$ the prior for the nuisance parameters under H_1 should equal the prior for the nuisance parameters under H_0. When this condition holds, the nuisance parameters can be ignored, so that again

$$BF_{01} = \frac{p(D|H_0)}{p(D|H_1)} = \frac{p(\phi = \phi_0|D, H_1)}{p(\phi = \phi_0|H_1)}, \tag{14.7}$$

which equals the ratio of the heights for the posterior and the prior distribution for ϕ at ϕ_0. Thus, the Savage-Dickey density ratio holds under relatively general conditions.

Equation 14.7 conveys several important messages:

1. **Relevance of the prior for the parameter of interest** The denominator of Equation 14.7 features the height of the prior for ϕ at $\phi = \phi_0$. This means that the choice of prior can greatly influence the Bayes factor, a fact that is also illustrated by Figures 14.1 and 14.2. The choice of prior will also influence the shape of the posterior, of course, but this influence quickly diminishes as the data accumulate. This point underscores the conceptual challenge for the Bayes factors that was noted earlier (e.g., Bartlett, 1957; Liu & Aitkin, in press). For example, consider again a test for a Normal mean μ, with $H_0 : \mu = 0$ and $H_1 : \mu \neq 0$. Suppose the prior for μ is a uniform distribution that ranges from $-a$ to a, and suppose that the number of observations is reasonably large. In this situation, the data will have overwhelmed the prior, so that the posterior for μ is relatively robust against changes in a. In contrast, the height of the prior at $\mu = 0$ varies directly with a: if a is doubled, the height of the prior at $\mu = 0$ becomes twice as small, and according to Equation 14.7 this would about double

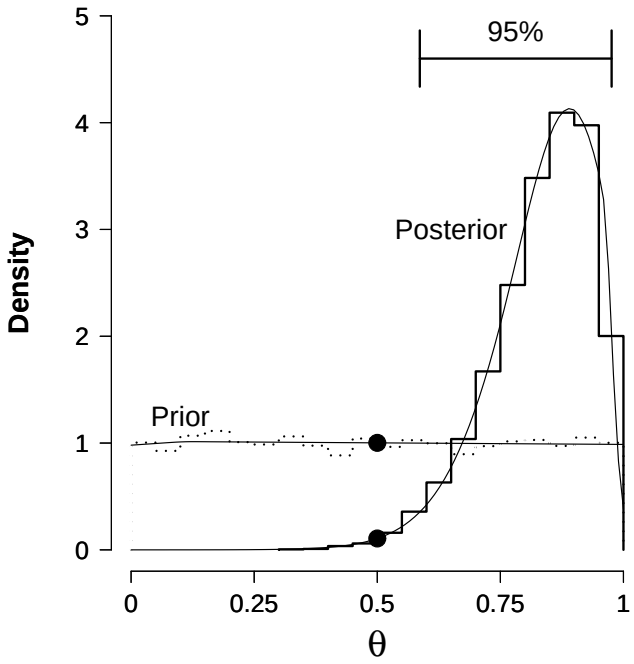


Figure 14.2: MCMC-based prior and posterior distributions for binomial rate parameter θ, after observing 9 correct responses and 1 incorrect response. The thin solid lines indicate the fit of a nonparametric logspline density estimator. Based on this density estimator, the mode of the posterior distribution for θ is approximately 0.89, and the 95% confidence interval extends from 0.59 to 0.98, closely matching the analytical results from Figure 14.1.

the Bayes factor in favor of H_0. In the limit, as a grows very large, the height of the prior at $\mu = 0$ goes to zero, which means that the Bayes factor will go to infinity, indicating decisive support for the null hypothesis.

2. **Irrelevance of the prior for nuisance parameters** In contrast to the prior for the parameter of interest ϕ, Equation 14.7 indicates that the prior for the nuisance parameters ψ is not critical. Hence, priors on the nuisance parameters can be vague or even improper (e.g., Hsiao, 1997, p. 659; Kass & Raftery, 1995, p. 783; Kass & Vaidyanathan, 1992). Intuitively, the prior vagueness of nuisance parameters is present in both models and cancels out in the computation of the Bayes factor (Rouder et al., 2009).

3. **Relative ease of computing the Bayes factor in nested models** Equation 14.7 shows that in nested models, under plausible assumptions on the prior structure for the nuisance parameters, computation of the Bayes factor is relatively straightforward. All that is needed is an estimate of posterior and prior ordinates under the alternative hypothesis H_1. This computational shortcut is often much less involved than the more generic solution, which involves integrating out nuisance parameters ψ for H_0, and parameters ψ and ϕ for H_1, as follows:

$$BF_{01} = \frac{p(D|H_0)}{p(D|H_1)} = \frac{\int p(D|\phi = \phi_0, \psi)p(\phi = \phi_0, \psi)d\psi}{\iint p(D|\psi, \phi)p(\psi, \phi)d\psi d\phi}. \tag{14.8}$$

The next chapters use concrete examples to illustrate how statisticians and cognitive scientists can use Bayesian hypothesis tests to their advantage.

CHAPTER 15

BAYESIAN T-TESTS

WITH RUUD WETZELS

Popular theories are difficult to overthrow. Consider, for instance, the following hypothetical sequence of events. First, Dr. John proposes a seasonal memory model (SMM). The model is intuitively attractive and quickly gains in popularity. Dr. Smith, however, remains unconvinced and decides to put one of SMMs predictions to the test. Specifically, SMM predicts that the increase in recall performance due to the intake of glucose is more pronounced in summer than in winter. Dr. Smith conducts the relevant experiment using a within subjects design and finds the exact opposite, although the result is not significant. More specifically, Dr. Smith finds that with $n = 41$ the t-value equals 0.79, which corresponds to a two-sided p-value of .44 (see Table 15.1).

Table 15.1: Increase in recall performance due to intake of glucose in summer and winter, $t = 0.79$, $p = .44$ (NB: hypothetical example).

Season	N	Mean	SD
Winter	41	0.11	0.15
Summer	41	0.07	0.23

Clearly, Dr. Smith's data do not support SMMs prediction that the glucose-driven increase in performance is larger in summer than in winter. Instead, the data seem to suggest that the null hypothesis is plausible, and that no difference between summer and winter is evident. Dr. Smith submits his findings to the *Journal of Experimental Psychology: Learning, Memory, and the Seasons*. Three months later, Dr. Smith receives the reviews, and one of them is from Dr. John. This review includes the following comment:

> "From a null result, we cannot conclude that no difference exists, merely that we cannot reject the null hypothesis. Although some have argued that with enough data we can argue for the null hypothesis, most agree that this is only a reasonable thing to do in the face of a sizeable amount [sic] of data [which] has been collected over many experiments that control for all concerns. These conditions are not met here. Thus, the empirical contribution here does not enable readers to conclude very much, and so is quite weak (...)."[1]

Formally, Dr. John's, first statement is completely correct: p-values cannot be used to quantify the support in favor of the null hypothesis. Here we present a Bayesian t-test, inspired by the test proposed by Rouder et al. (2009), that can be used to quantify such support. More information can be found on the website of Ruud Wetzels, `http://www.ruudwetzels.com` .

15.1 A One-Sample t-Test

When we use the one-sample t-test, we assume that the data are Normally distributed with unknown mean μ and unknown variance σ^2. In the within-subjects experiment of Dr. Smith, the data consist of the standardized difference scores. The null hypothesis states that the mean of these data is equal to zero, that is, $H_0 : \mu = 0$. The alternative hypothesis states that the mean is not equal to zero, that is, $H_1 : \mu \neq 0$.

We follow Rouder et al. and use a Cauchy(0,1) prior for effect size δ. The Cauchy distribution is a t-distribution with 1 degree of freedom; compared to the Normal distribution, the Cauchy distribution has fatter tails (cf. Figure 15.1). The advantage of defining a prior on effect size (instead of on the mean) is that it is very general; the same prior can be used across many experiments, dependent variables, and measurement scales.

[1] This quote is taken from an actual review.

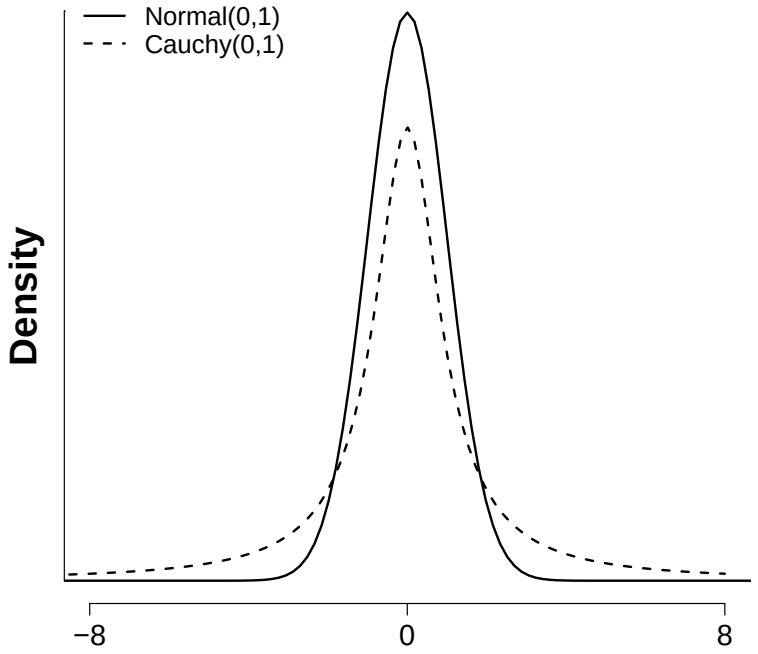


Figure 15.1: Comparison between the standard Normal distribution and the standard Cauchy distribution. The Cauchy distribution has relatively fat tails.

For the standard deviation σ we use a half-Cauchy(0,1) (Gelman & Hill, 2007), that is, a Cauchy(0,1) distribution that is defined only for positive numbers. This choice for σ is reasonably uninformative. The graphical model for the one–sample t–test is shown in Figure 15.2.

In the graphical model, X represents the observed data, distributed according to a Normal distribution with mean μ_X and a variance σ_X^2. Effect size δ is defined as $\delta = \mu_X/\sigma_X$, and so μ_X is given by $\mu_X = \delta \times \sigma_X$. The null hypothesis puts all prior mass for δ on a single point, that is, $H_0 : \delta = 0$, whereas the alternative hypothesis assumes that δ is Cauchy(0,1) distributed, $H_1 : \delta \sim$ Cauchy(0,1).

The following code implements the graphical model in WinBUGS.

```
model
{
# Data are Normally Distributed
  for (i in 1:nsubj)
  {
    X[i] ~ dnorm(muX,lambdaX)
  }
  lambdaX <- pow(sigmaX,-2)

# Delta and sigmaX come from a Cauchy distribution,
# which is not predefined in WinBUGS. A common
# solution is to draw delta and sigmaX from
# from a Normal distribution with mean 0
# and precision given by a Chi Square
# distribution with 1 degree of freedom.

  delta       ~ dnorm(0,lambdaDelta)
  lambdaDelta ~ dchisqr(1)
```

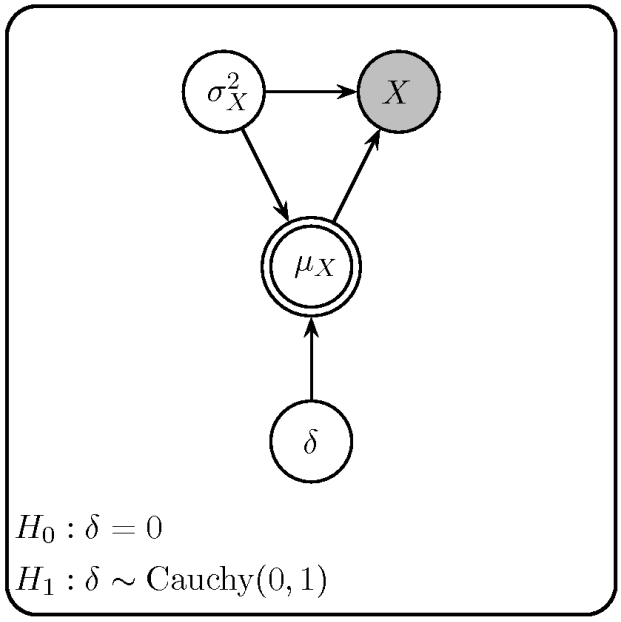


Figure 15.2: Graphical model for the SD one–sample t–test. Cauchy(0,1)$^+$ denotes the half–Cauchy(0,1) defined for positive numbers only.

```
  sigma     ~ dnorm(0,lambdaChi)
  lambdaChi ~ dchisqr(1)

# Sigma can only be positive (half-Cauchy)
  sigmaX <- abs(sigma)

# delta = muX/sigmaX:
  muX <- delta * sigmaX
}
```

The code `tTest_1.m` (Matlab) or `tTest_1.R` (R) calls WinBUGS to draw samples from the posterior for effect size δ. Next, all that is required to compute the Bayes factor is the height of the prior and posterior distributions for δ at $\delta = 0$. The height of the prior distribution at $\delta = 0$ can be immediately computed from the Cauchy(0,1) distribution. The height of the posterior distribution at $\delta = 0$ can be easily estimated from the MCMC samples, for instance by applying a nonparametric density estimator (e.g., Stone et al., 1997).

In R, this estimator is included in the polspline package. To install the `polspline` package, start R and select the `Install Package(s)` option in the `Packages` menu. Once you chose your preferred CRAN mirror (e.g., Netherlands (Amsterdam 2)), select `polspline` in the `Packages` window and click on `OK`. In Matlab, you may try the nonparametric density estimators available in the free package developed by Zdravko Botev (`http://www.mathworks.com/matlabcentral/fileexchange/14034`).

The code `tTest_1.R` also plots the prior and the posterior distributions for δ. Negative values of δ indicate that the effect of glucose is larger in summer than in winter, as is predicted by SMM. The top panel of Figure 15.3 shows the results for the Bayes factor for $H_0 : \delta = 0$ versus the unrestricted alternative $H_1 : \delta \neq 0$, instantiated as $\delta \sim$ Cauchy(0,1). This test shows that the data are about six times more likely under H_0 than under H_1.

However, this was not the hypothesis that Dr. Smith set out to test. This hypothesis specifically stated that δ should be negative. The middle panel of Figure 15.3 shows the results for the associated Bayesian t-

test, that is, $H_0 : \delta = 0$ versus the order-restricted hypothesis $H_1 : \delta < 0$, instantiated as $\delta \sim \text{Cauchy}(0,1)^-$, a half-Cauchy(0,1) distribution that is defined only for negative numbers. In order to calculate the height of the order-restricted posterior distribution at $\delta = 0$, we focus solely on that part of the unrestricted posterior for which $\delta < 0$. After renormalizing, we obtain a truncated but proper posterior distribution that ranges from $\delta = -\infty$ to $\delta = 0$. Figure 15.3 shows both the half-Cauchy(0,1) prior (solid line) and the truncated posterior (dashed line). The Savage-Dickey ratio at $\delta = 0$ yields a Bayes factor of $BF_{01} = 13.75$. This means that the data are almost 14 times more likely under H_0 than under the order-restricted H_1 that is associated with SMM. When H_0 and H_1 are equally likely *a priori*, the posterior probability in favor of the null hypothesis is about $13.75/14.75 \approx .93$, which is considered "positive evidence" for the null hypothesis (Raftery, 1995; Wagenmakers, 2007).

For completeness, the bottom panel of Figure 15.3 shows the Bayesian t-test for the alternative order-restriction. In this case, we seek to test $H_0 : \delta = 0$ versus $H_1 : \delta > 0$, instantiated as $\delta \sim \text{Cauchy}(0,1)^+$, a half-Cauchy(0,1) distribution that is defined only for positive numbers. The Savage-Dickey density ratio yields a Bayes factor of $BF_{01} = 3.91$, which indicates that the data are almost 4 times more likely under H_0 than under H_1.

Exercises

1. Here we assumed a half-Cauchy prior distribution on the variance `sigmaX`. Other choices are possible and reasonable. Can you think of a few?
2. Do you think the different priors on choices on `sigmaX` will lead to substantially different conclusions? Why or why not? Convince yourself by implementing a different prior and study the result.
3. We also assumed a Cauchy prior distribution on effect size `delta`. Other choices are possible and reasonable. One such choice is the standard Normal distribution. Do you think this prior will lead to substantially different conclusions? Why or why not? Convince yourself by implementing the standard Normal prior and study the result.
4. In this example, it matters greatly whether H_1 is unrestricted, order-restricted to negative values for δ, or order-restricted to positive values for δ (cf. the different panels in Figure 15.3). Can you intuit why this is perfectly reasonable? Can you think of a situation where the three versions of H_1 yield exactly the same Bayes factor?

15.2 A Two-Sample t-Test

The two–sample t–test is used to test whether the population means of two independent samples of observations are equal to each other or not. In experimental psychology, the two–sample t–test is often used for between–subjects designs.

The input for the one-sample t-test were standardized difference scores; for the two-sample t-test, we use rescaled data as input. Specifically, we rescale the data such that one group has mean 0 and standard deviation 1. This scaling does not affect the test statistic. For the data from Dr. Smith, for instance, the "summer mean" of 0.07 is subtracted from all observations, both in the winter condition and in the summer condition. Next, all observations are divided by the "summer standard deviation". The main advantage of this rescaling procedure is that the prior distributions for the parameters hold regardless of the scale of measurement: for our Bayesian t-test, it does not matter whether, say, response times are measured in seconds or in milliseconds.

The graphical model for the two–sample t–test is shown in Figure 15.4. Nodes X and Y represent the two groups of observed data. Both X and Y are distributed according to a Normal distribution with shared variance σ^2. The mean of X is given by $\mu + \alpha/2$, and the mean of Y is given by $\mu - \alpha/2$, which means that α is the difference in the means.

Because $\delta = \alpha/\sigma$, α is given by $\alpha = \delta \times \sigma$. As for the one–sample scenario, the null hypothesis puts all prior mass for δ on a single point, that is, $H_0 : \delta = 0$, whereas the alternative hypothesis assumes that δ is Cauchy(0,1) distributed, $H_1 : \delta \sim \text{Cauchy}(0,1)$.

The following code implements the graphical model in WinBUGS.

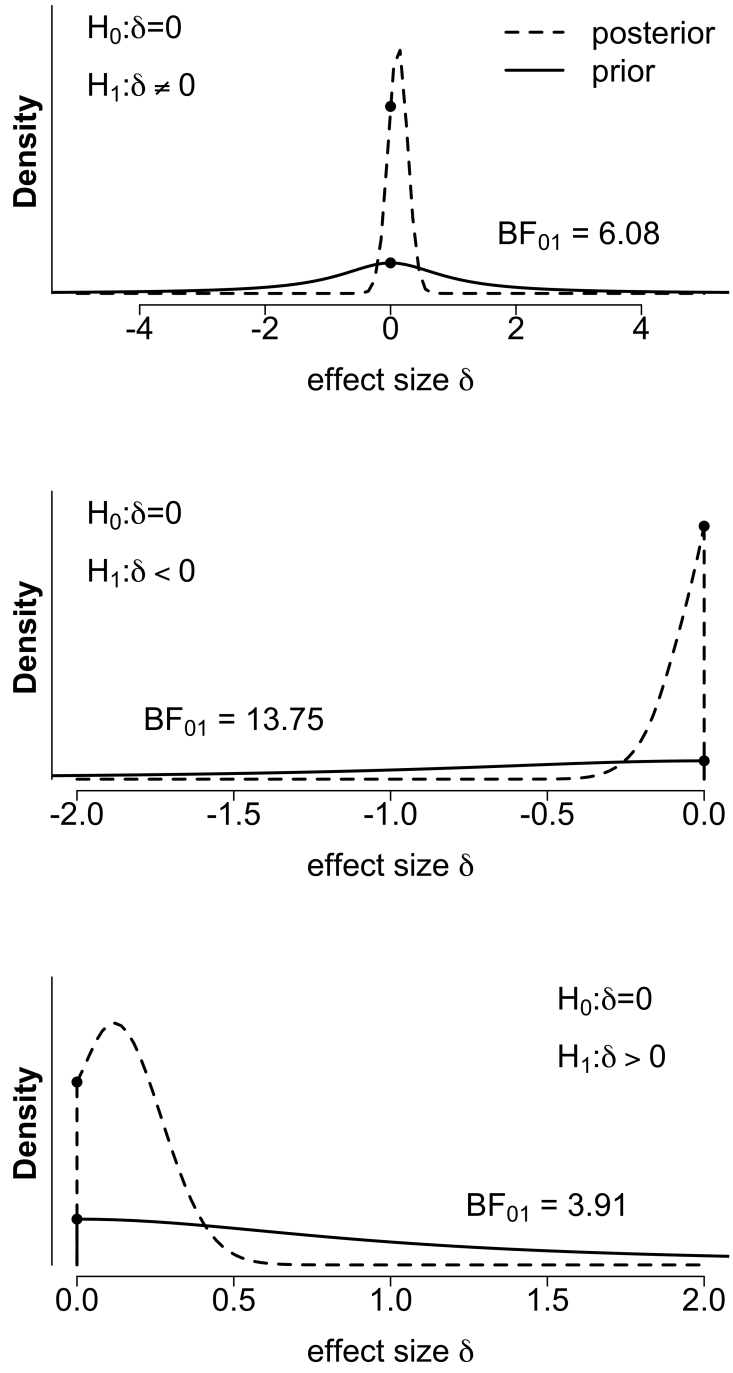
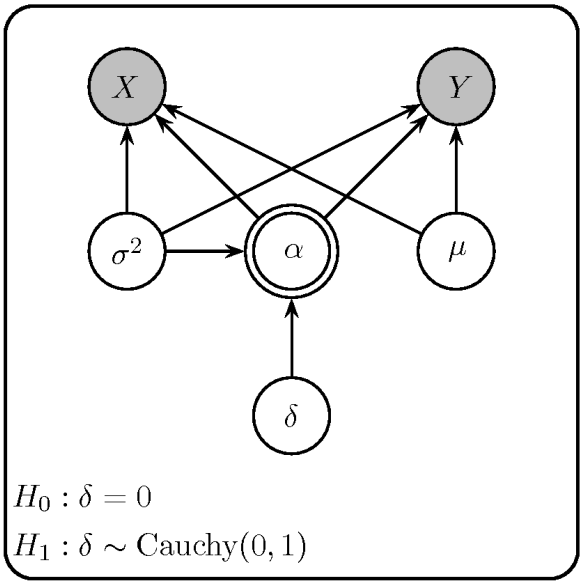


Figure 15.3: The prior and posterior distributions of effect size δ, based on the data from Dr. Smith. The top panel illustrates the unrestricted t-test, the middle panel illustrates the order-restricted test associated with the SMM, and the bottom panel illustrates the t-test for the alternative order-restriction. The dots mark the height of the prior and posterior distributions at δ=0.



$$X \sim \text{Normal}(\mu + \alpha/2, \sigma^2)$$
$$Y \sim \text{Normal}(\mu - \alpha/2, \sigma^2)$$

$$\sigma \sim \text{Cauchy}(0,1)^+$$
$$\mu \sim \text{Cauchy}(0,1)$$
$$\alpha = \sigma \times \delta$$

$$\delta \sim \text{Cauchy}(0,1)$$

Figure 15.4: Graphical model for the SD two–sample t–test. Cauchy(0,1)$^+$ denotes the half–Cauchy(0,1) defined for positive numbers only.

```
model
{
  for (i in 1:n1)
  {
    group1[i] ~ dnorm(muX,lambdaXY)
  }
  for (i in 1:n2)
  {
    group2[i] ~ dnorm(muY,lambdaXY)
  }

  lambdaXY <- pow(sigmaXY,-2)

  delta       ~ dnorm(0,lambdaDelta)
  lambdaDelta ~ dchisqr(1)

  sigma    ~ dnorm(0,sigmaChi)
  sigmaChi ~ dchisqr(1)
  sigmaXY <- abs(sigma)

  mu    ~ dnorm(0,muChi)
  muChi ~ dchisqr(1)

  alpha <- delta*sigmaXY

  muX <- mu + alpha*0.5
  muY <- mu - alpha*0.5
}
```

Exercises

1. Imagine that you would apply the two-sample test to the data from Dr. Smith, pretending that the experiment was between-subjects instead of within-subjects. How do you think the results would change?
2. Test your intuition by running the code `tTest_2.R`.
3. The two-sample *t*-test outlined above assumes that the two groups have equal variance. How would you extend the present approach to deal with this complication?

CHAPTER 16

HYPOTHESIS TESTS INVOLVING BINOMIAL DISTRIBUTIONS

16.1 Equality of Proportions

In their article "After the promise: the STD consequences of adolescent virginity pledges", Brückner and Bearman (2005) analyzed a series of interviews conducted as part of the National Longitudinal Study of Adolescent Health (*Add Health*). The focus of the article was on the sexual behavior of adolescents, aged 18-24, who have made a *virginity pledge*, that is, a public or written pledge to remain a virgin until marriage. Scientific studies suggest that the sexual behavior of pledgers is not very different from that of nonpledgers—except for the fact that pledgers are less likely to use condoms when they first have sex.

The Brückner and Bearman (2005) study presents a wealth of data, but here our focus is on a small subset of the data: 424 out of 777 pledgers ($\approx 54.6\%$) indicated that they had used a condom at first sex, versus 5416 out of 9072 nonpledgers ($\approx 59.7\%$). To what extent does a statistical analysis support the assertion that pledgers are less likely than nonpledgers to use a condom at first sex?

A frequentist test for equality of proportions indicates that $p \approx .006$, which tells us that when H_0 is true (i.e., the proportions of condom users are equal in the two groups), then the probability is about .006 that we would encounter a result at least as extreme as the one that was in fact observed. But this is not the kind of information that researchers really care about; researchers want to know the extent to which the data support the claim that pledgers are less likely than nonpledgers to use a condom at first sex.

Our Bayesian model for these data is simple and general. We assume that the number of condom users ($s_1 = 424$ and $s_2 = 5416$) among the pledgers and the nonpledgers ($n_1 = 777$ and $n_2 = 9072$) is governed by binomial rate parameters θ_1 and θ_2, respectively. Denote the difference between the two rate parameters by δ, that is, $\delta = \theta_1 - \theta_2$. Figure 16.1 shows this model in graphical model notation.

In our Bayesian model, we assume that the rate parameters θ_1 and θ_2 each have a uniform prior distribution (i.e., $p(\theta_{(\cdot)}) \sim \text{Beta}(1, 1)$). These uniform prior distributions induce a triangular prior distribution

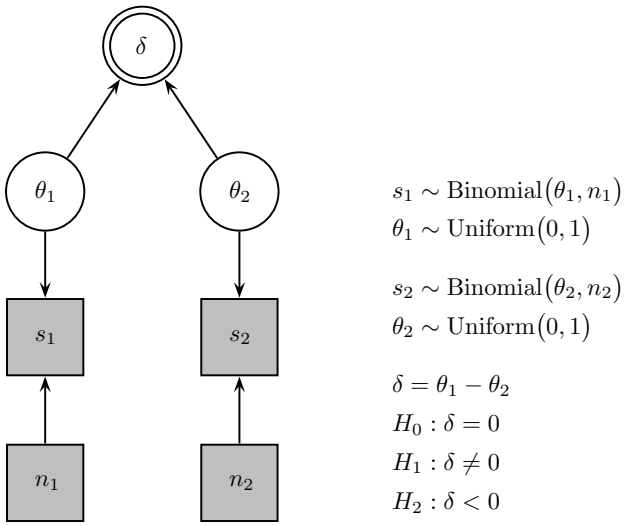


Figure 16.1: Bayesian graphical model for the pledger data.

for the difference parameter δ:

$$p(\delta) = \begin{cases} 1+\delta & \text{for } \delta \leq 0, \\ 1-\delta & \text{for } \delta > 0. \end{cases} \tag{16.1}$$

The null hypothesis states that the rates θ_1 and θ_2 are equal, and hence $H_0 : \delta = 0$. The unrestricted alternative hypothesis states that the rates are free to vary, $H_1 : \delta \neq 0$, and the restricted alternative hypothesis states that the rate is lower for the pledgers than for the nonpledgers, $H_2 : \delta < 0$. Below we examine these alternative hypothesis in turn.

Unrestricted Analysis

The problem of testing $H_0 : \delta = 0$ versus $H_1 : \delta \neq 0$ is still relatively simple. The Bayes factor in support for the null hypothesis (i.e., $BF_{01} = p(D|H_0)/p(D|H_1)$) is given for instance by:

$$BF_{01} = \frac{\binom{n_1}{s_1}\binom{n_2}{s_2}}{\binom{n_1+n_2}{s_1+s_2}} \frac{(n_1+1)(n_2+1)}{n_1+n_2+1}. \tag{16.2}$$

For the pledger data, this yields $BF_{01} \approx 0.45$, which means that the data are about $1/0.45 \approx 2.22$ times more likely under the alternative hypothesis than under the null hypothesis. Note that although the Bayesian hypothesis test supports the alternative hypothesis, the result is much less convincing than a p-value of .006 suggests.

To apply the Savage-Dickey test, we first draw samples from the posterior and the prior distributions for δ. The following code implements the graphical model in WinBUGS.

```
model
{
  # Uniform Prior on Rates:
  theta1 ~ dbeta(1,1)
  theta2 ~ dbeta(1,1)

  # Binomial Distribution for Observed Counts:
  s1 ~ dbin(theta1,n1)
  s2 ~ dbin(theta2,n2)

  # Difference between Rates:
  delta <- theta1-theta2

  # Priors
  # Make "Dummy" Variables That Copy The Prior,
  # But Are Never Updated By Data
  theta1prior ~ dbeta(1,1)
  theta2prior ~ dbeta(1,1)
  deltaprior <- theta1prior-theta2prior
}
```

The code `Pledgers_1.m` (Matlab) or `Pledgers_1.R` (R) calls WinBUGS to draw samples from the posterior and the prior for the rate difference δ. The left panel of Figure 16.2 shows the resulting histograms on their entire range. In this panel, the thin solid line for the prior indicates the analytical distribution given in Equation 16.1. For the posterior distribution, the thin solid line indicates a logspline nonparametric density estimate (Stone et al., 1997).

The right panel of Figure 16.2 zooms in on the relevant region around $\delta = 0$. The almost flat line is the analytical distribution of the prior, and the sharply decreasing line is the logspline estimate for the posterior.

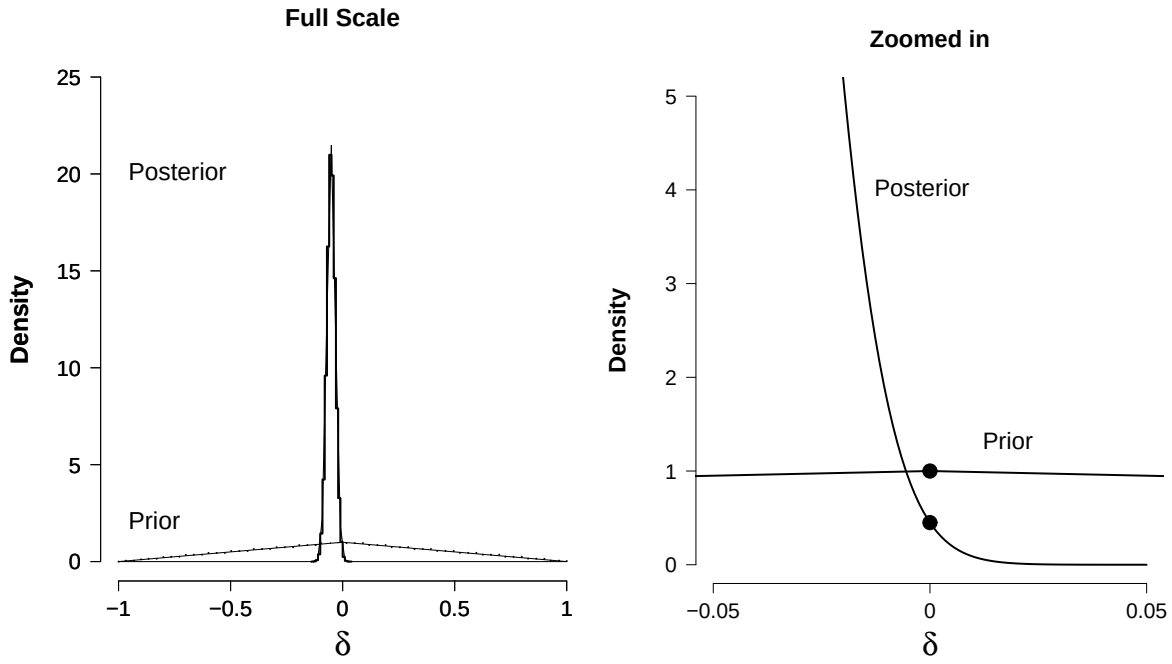


Figure 16.2: Prior and posterior distributions of the rate difference δ for the unrestricted analysis of the pledger data. The left panel shows the distributions across their entire range (prior: histogram and analytical result; posterior: histogram and logspline density estimate). The right panel zooms in on the area that is relevant for the Savage-Dickey test of $H_0 : \delta = 0$ versus $H_1 : \delta \neq 0$ (prior: analytical result; posterior: logspline density estimate). The dots indicate the height of the two distributions at $\delta = 0$.

The two dots mark the height of both densities at $\delta = 0$. From a visual comparison of the height of the dots, it is clear that the point $\delta = 0$ is supported about twice as much under the prior as it is under the posterior. That is, the data have decreased the support for $\delta = 0$ by a factor of two. Application of the Savage-Dickey procedure yields $BF_{01} \approx 0.47$, which leads to the conclusion that the data are about 2.17 times more likely under the alternative hypothesis than under the null. Thus, the result from the MCMC-based Savage-Dickey test (i.e., $BF_{10} = 2.17$) and the analytical solution (i.e., $BF_{10} = 2.22$) are in reasonable agreement.

Finally, note that the conclusions from the Bayesian hypothesis test (i.e., roughly twice as much evidence for H_1 as for H_0) are more conservative than those that follow from Bayesian parameter estimation; the Bayesian 95% confidence interval for the posterior of δ is $(-0.09, -0.01)$ and does not include 0. The reason for the discrepancy is that the Bayesian hypothesis test punishes H_1 for assigning prior mass to values of δ that yield very low likelihoods (i.e., the automatic Ockham's razor discussed previously, see Berger & Delampady, 1987 for a discussion).

Exercises

1. In the current analysis, we put independent priors on θ_1 and θ_2. Do you think this is plausible? How would you change the model to take into account the possible dependence? How would this affect the outcome of the Bayesian test?

Order-Restricted Analysis

Many substantive psychological questions can be formulated as order-restrictions (e.g., Hoijtink, Klugkist, & Boelen, 2008). Here we focus on a test of $H_0 : \delta = 0$ versus $H_2 : \delta < 0$, an order-restricted alternative

hypothesis that states that the rate of condom use is lower for the pledgers than for the nonpledgers.

In the Bayesian framework, order-restrictions can be implemented in several ways (e.g., O'Hagan & Forster, 2004, pp. 70-71). For instance, order-restrictions can be enforced before MCMC sampling, by appropriately constraining the prior distributions, or they can be implemented after the MCMC sampling, by retaining only those MCMC samples that obey the order-restriction.

The left panel of Figure 16.3 shows the histograms for the posterior and prior distributions for δ under the restricted model $H_2 : \delta < 0$. These histograms were obtained by selecting from the previous unrestricted analysis only those MCMC samples that obey the order-restriction. For the prior, the thin solid line indicates the analytical distribution, and for the posterior it indicates the order-restricted logspline estimate.

Note that for the prior, the effect of the order-restriction is to double the mass on $\delta = 0$, from a value of 1 to a value of 2. In contrast, the order-restriction does not much affect the posterior, as most of its mass was already smaller than 0. The right panel of Figure 16.3 zooms in on the relevant area around $\delta = 0$ and shows the effect of the order-restriction on the Bayesian hypothesis test. Again, the almost flat line is the analytical distribution of the order-restricted prior, and the associated dot indicates its height at $\delta = 0$. The sharply decreasing line is the logspline estimate for the order-restricted posterior, and the associated solid dot indicates the logspline estimate of the height of the posterior based on the subset of MCMC samples that obey the order-restriction. The open dot immediately below indicates the height of the posterior estimated from an alternative method, one that is based on renormalizing the order-restricted posterior (i.e., dividing the height of the unrestricted posterior at $\delta = 0$ by the area of the unrestricted posterior that lies to the left of $\delta = 0$). Both methods are illustrated in the code `Pledgers_2.R` (R).

A visual comparison of the height of the prior and posterior at $\delta = 0$ confirms that the order-restriction has increased the evidence in favor of the alternative hypothesis. Specifically, the logspline estimate yields $BF_{02} \approx 0.26$ (i.e., $BF_{20} \approx 3.78$), and the estimate based on renormalizing the posterior yields $BF_{02} \approx 0.23$ (i.e., $BF_{20} \approx 4.34$). Thus, both methods lead to the conclusion that there is roughly four times as much evidence for H_2 as for H_0.

The foregoing may lead one to conclude that the effect of order-restrictions are similar in the Bayesian and the frequentist framework; in the Bayesian framework, the order-restriction increased the evidence against H_0 roughly by a factor of two, and in the frequentist framework, a one-sided p-value provides twice as much evidence against H_0 as a two-sided p-value. However, this correspondence only holds because the posterior for δ is largely consistent with the order-restriction, as the next exercise confirms.

Exercises

1. For the pledger data, consider an order-restricted test of $H_0 : \delta = 0$ versus $H_3 : \delta > 0$. What do you think the result will be? Check your intuition by adjusting the code `Pledgers_2.R` to carry out the required test.

16.2 A Hierarchical Bayesian One-Sample t-Test

In their article "Priming in implicit memory tasks: Prior study causes enhanced discriminability, not only bias", Zeelenberg, Wagenmakers, and Raaijmakers (2002) reported three experiments in two-alternative forced-choice perceptual identification. In the test phase of each experiment, a stimulus (e.g., a picture of a clothes pin) is briefly presented and masked. Immediately after the mask the participant is confronted with two choice options—the target (i.e., the picture of the clothes pin) and a similar foil alternative (e.g., the picture of a stapler; see Figure 16.4 for an example); the participant's goal is to identify the target.

Prior to the test phase, the Zeelenberg et al. experiments featured a study phase, in which participants studied a subset of the choice alternatives that would also be presented in the later test phase. Two conditions were critical: the "study-neither" condition, in which neither choice alternative was studied, and the "study-both" condition, in which both choice alternatives were studied.

In the first two experiments reported by Zeelenberg et al., participants choose the target stimulus more often in the study-both condition than in the study-neither condition. This *both-primed benefit* suggests that prior study leads to enhanced discriminability, not just a bias to prefer the studied alternative (e.g., Ratcliff & McKoon, 1997).

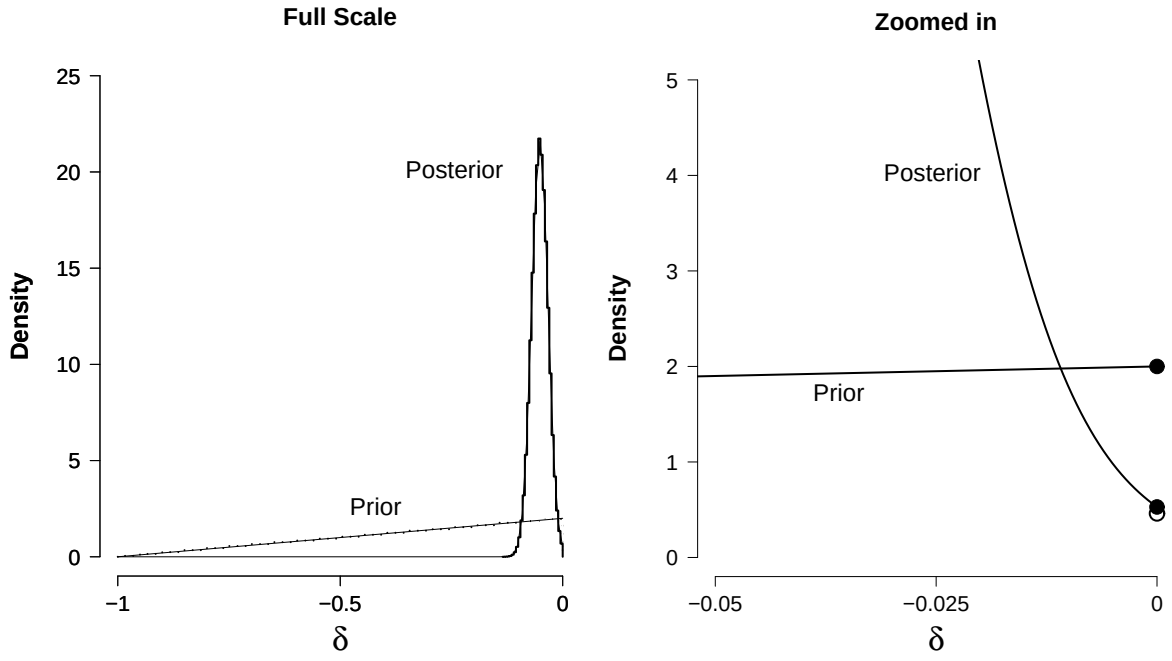


Figure 16.3: Prior and posterior distributions of the rate difference δ for the order-restricted analysis of the pledger data. The left panel shows the distributions across their entire range (prior: histogram and analytical result; posterior: histogram and logspline density estimate). The right panel zooms in on the area that is relevant for the Savage-Dickey test of $H_0 : \delta = 0$ versus $H_2 : \delta < 0$ (prior: analytical result; posterior: logspline density estimate). The dots indicate the height of the two distributions at $\delta = 0$.

Figure 16.4: Example pair of similar pictures used in Experiment 3 from Zeelenberg et al. (2002).

Here we focus on statistical inference for the Experiment 3 from Zeelenberg et al. (2002). In the study phase of this experiment, all 74 participants were presented with 21 pairs of similar pictures (e.g., the clothes pin/stapler example shown in Figure 16.4). In the test phase, all participants had to identify briefly presented target pictures among a set of two alternatives. The test phase was composed of 42 pairs of similar pictures, 21 of which had been presented in the study phase.

In order to assess the evidence in favor of the both-primed benefit, the authors carried out a standard analysis and computed a one-sample t-test:

> "Mean percentage of correctly identified pictures was calculated for each participant. When neither the target nor the foil had been studied, 71.5% of the pictures were correctly identified. When both the target and the foil had been studied, 74.7% of the pictures were correctly identified. The difference between the study-both and study-neither conditions was significant, $t(73) = 2.19$, $p < .05$."

Our Bayesian test of the both-primed benefit proceeds as follows. We start by assuming that for participant i the number of correct choices is binomially distributed with parameter θ_i. Unfortunately, θ_i is defined on the rate scale, which ranges from 0 to 1. This is an awkward scale for modeling additive effects, as a change from .55 to .65 is not the same as a change from .85 to .95. Hence, we do not model θ_i, but instead choose to model ϕ_i, the probit transformation of θ_i.

The probit transform is the inverse cumulative distribution function of the standard Normal distribution, so that, for instance, a rate of $\theta_i = 0.5$ maps onto a probit rate of $\phi_i = 0$, and a rate of $\theta_i = 0.975$ maps onto a probit rate of $\phi_i = 1.96$ (see Figure 7.2). In contrast to the rate scale, the probit scale covers the entire real line, and lends itself easily to hierarchical modeling.

For each participant i, the both-primed benefit α_i is given by the difference between performance in the study-both and study-neither condition, $\alpha_i = \phi_{SB,i} - \phi_{SN,i}$. Our model incorporates two random effects; first, each participant's baseline level of performance $\phi_{SN,i}$ is assumed to be drawn from a group-level Normal distribution with mean μ_ϕ and standard deviation σ_ϕ. Second, each participant's both-primed benefit is assumed to be drawn from a group-level Normal distribution with mean μ_α and standard deviation σ_α. Note that such Normal distributions are easily defined on the probit scale, but not on the rate scale. Figure 16.5 shows the model in graphical form. Because each participant contributes to both the study-neither and the study-both conditions, the design is "within-subjects" and the subject plate therefore encloses both conditions.

For the parameters that are not subject to statistical test (i.e., μ_ϕ, σ_ϕ, and σ_α) we specified uninformative priors. The prior for the group mean of the study-neither condition, μ_ϕ, is a truncated standard Normal (i.e., greater than zero only on the positive real line), which on the rate scale translates to a uniform distribution from 0.5 to 1. For σ_ϕ and σ_α, we chose priors that are uniform from 0 to 10.

Finally, and critically, our model incorporates a parameter δ that quantifies *effect size*, $\delta = \mu_\alpha/\sigma_\alpha$. Effect size is a dimensionless quantity, and this makes it relatively easy to define a principled prior (cf. the earlier chapter on Bayesian t-tests). Reasonable default choices for priors on effect size include the Cauchy distribution (i.e., a t distribution with one degree of freedom) and the standard Normal distribution (e.g., Gönen, Johnson, Lu, & Westfall, 2005; Rouder et al., 2009). The latter prior is known as the "unit information prior", as it carries as much information as a single observation (Kass & Wasserman, 1995). The standard Normal distribution is the prior for effect size that we will use in this example and the next.

With the statistical model in place, we can now turn to hypothesis testing. The null hypothesis states that there is no both-primed benefit, and hence the effect size is zero: $H_0 : \delta = 0$. The alternative, order-restricted hypothesis states that there is a both-primed benefit, and hence $H_1 : \delta > 0$. This test is, in fact, a hierarchical extension of the Bayesian one-sample t-test discussed in the previous chapter.

The following code implements the graphical model in WinBUGS.

```
model
{
  for(i in 1:74) # 74 Participants
  {
    # Binomial Distributions for Observed Counts:
    KSN[i] ~ dbin(thetaSN[i],NSN[i])
    KSB[i] ~ dbin(thetaSB[i],NSB[i])
```

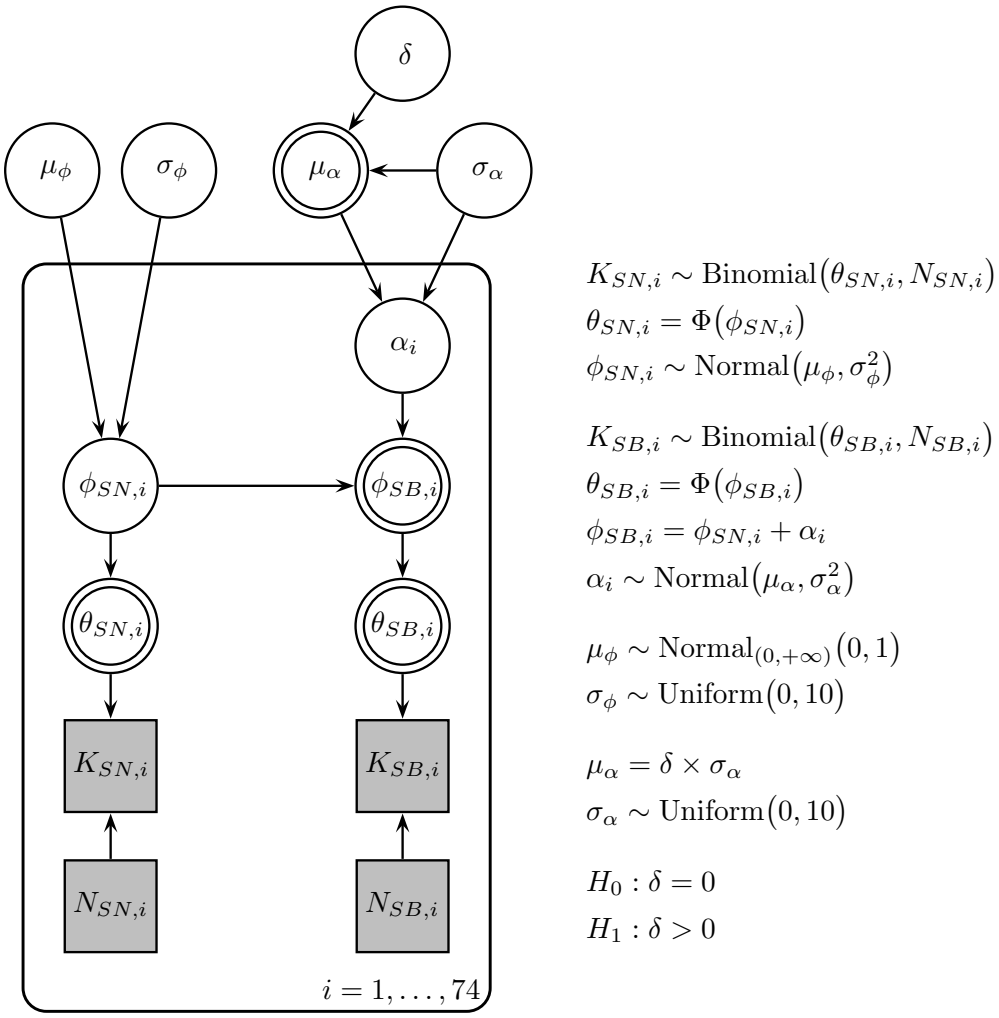


Figure 16.5: Bayesian graphical model for the Zeelenberg data. In a within-subjects design, 74 participants performed a two-alternative forced-choice perceptual identification task, in both "study-neither" (SN) and "study-both" (SB) conditions.

```
# Transformation to Parameters on the Probit Scale:
thetaSN[i] <- phi(phiSN[i])
thetaSB[i] <- phi(phiSB[i])

# Individual Parameters that Quantify Performance in
# the Study-Neither Condition Come From a Group-Level Distribution:
phiSN[i] ~ dnorm(muphi,tauphi)
# NB. tauphi is the precision, defined as 1/variance

# On the Probit Scale, Priming Effects Are Additive:
phiSB[i] <- phiSN[i]+alpha[i]
# alpha[i] is the priming effect for participant i

# Individual Priming Effects Come From a Group-Level Distribution:
```

```
    alpha[i] ~ dnorm(mualpha,taualpha)
    # NB. taualpha is the precision, defined as 1/variance
  }

  # Group-Level Priors for the Study-Neither Condition:
  muphi ~  dnorm(0,1)I(0,)
  # NB1. The I(0,) command ensures that all samples for muphi are > 0
  # NB2. This prior for muphi corresponds to a uniform prior the rate scale,
  # ranging from 0.5 to 1.

  # Uninformative Prior on the Group-Level Standard Deviation:
  sigmaphi ~  dunif(0,10)
  # Transformation from Standard Deviation to Precision:
  tauphi <- pow(sigmaphi,-2)

  # Priors for the Group-Level Priming Effect (cf. Rouder et al., PBR):
  mualpha <- delta * sigmaalpha
  # Uninformative Prior for sigmaalpha:
  sigmaalpha ~ dunif(0,10)
  # Transformation from Standard Deviation to Precision:
  taualpha <- pow(sigmaalpha,-2)

  # The "Unit Information Prior" on Effect Size delta (cf. Rouder et al., PBR):
  delta ~ dnorm(0,1)I(0,)
  # NB. The I(0,) incorporates the order-restriction that allows only
  # positive values for delta
}
```

We implemented our Bayesian hierarchical t-test by means of the Savage-Dickey procedure. The code `Zeelenberg.m` (Matlab) or `Zeelenberg.R` (R) calls WinBUGS to draw samples from the posterior and the prior for the rate difference δ, and Figure 16.6 visualizes the results—for the prior on effect size δ, the thin solid line indicates the Normal distribution that has been truncated and renormalized to take into account the order restriction that $\delta > 0$. For the posterior order-restricted distribution on effect size δ, the thin solid line indicates the logspline nonparametric density estimate, and the thick solid line indicates the histogram of MCMC samples. As in the previous example, the two dots mark the height of prior and posterior densities at $\delta = 0$. From a visual comparison of the height of the dots, it is clear that the point $\delta = 0$ is supported about four times as much under the prior as it is under the posterior. That is, the data have decreased the support for $\delta = 0$ by a factor of four. Application of the Savage-Dickey test (i.e., Equation 14.7) yields $BF_{01} \approx 0.22$, which leads to the conclusion that the data are about 4.49 times more likely under the alternative hypothesis than under the null hypothesis.

Thus, the data support the assertion that there is a both-primed benefit, but the extent of this support is somewhat weaker than is suggested by the p-value.

Exercises

1. The Zeelenberg data can also be analyzed using the Bayesian t-test discussed in the previous chapter. Think of a few reasons why this might not be such a good idea. Then, despite your reservations, apply the Bayesian t-test anyway. Do the results differ? How? Why?

16.3 A Hierarchical Bayesian Two-Sample t-Test

In their article "How specific are executive functioning deficits in Attention Deficit Hyperactivity Disorder and autism?", Geurts, Verté, Oosterlaan, Roeyers, and Sergeant (2004) studied the performance of children

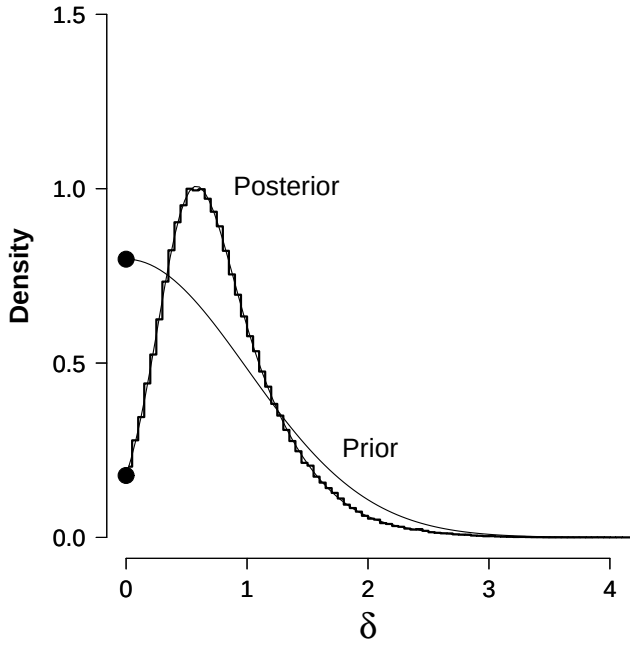


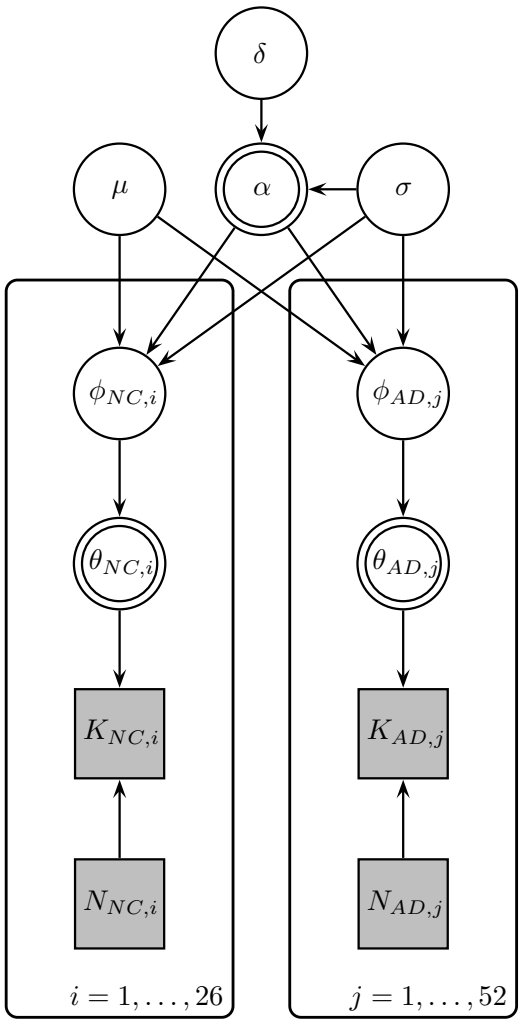
Figure 16.6: Prior and posterior distribution of the effect size δ for the hierarchical, order-restricted analysis of the Zeelenberg data. For the prior, the thin line gives the analytical result; for the posterior, the thick line gives the histogram and the thin line gives the logspline density estimate. The dots indicate the height of the two distributions at $\delta = 0$.

with ADHD and autism on a range of cognitive tasks. Here we focus on a small subset of the data and consider the question whether children that develop typically (i.e., "normal controls") outperform children with ADHD on the Wisconsin Card Sorting Test (WCST). The WCST requires that participants learn, by trial and error, to sort cards according to an implicit rule. The complication is that, over the course of the experiment, the sorting rule sometimes changes. This means that in order to avoid too many mistakes, participants have to suppress the tendency to perseverate and quickly discover and adopt the new rule. Because of these task demands, performance on the WCST is thought to quantify cognitive flexibility or set shifting ability.

The experiment of interest contains data from 26 normal controls and 52 children with ADHD. Each child performed the WCST, and the measure of interest is the number of correctly sorted cards relative to the total number of sorting opportunities. The WCST provides a maximum of 128 cards to sort, but, depending on a child's performance, this number could also be lower. Overall, the group of normal controls sorted the cards correctly on 65.4% of the cases, and the group of ADHD children sorted the cards correctly on 66.9% of the cases. A between-subjects (i.e., two-sample) frequentist t-test on the proportion of correctly sorted cards does not allow one to reject the null hypothesis, $t(40.2) = 0.37$, $p = .72$. But this statistic does not quantify the evidence in favor of the null hypothesis. Another problem with this frequentist t-test is that it ignores the fact that trials are nested in participants—a design that, as in the previous example, calls for a hierarchical/multi-level/random effects analysis.

Our hierarchical model is specified as follows. We assume that for child i in the group of normal controls, the number of correctly sorted cards $K_{NC,i}$ (out of $N_{NC,i}$ opportunities) is binomially distributed with rate parameter $\theta_{NC,i}$. As in the previous example, this rate parameter is then transformed to the probit scale, which yields the corresponding parameter $\phi_{NC,i}$. The comparable assumptions are made for child j in the group of ADHD children, resulting in the associated parameter $\phi_{AD,j}$.

Next, our model incorporates random effects; for both the normal controls and the group of ADHD children, the probitized rates of correct responding (i.e., $\phi_{NC,\cdot}$ and $\phi_{AD,\cdot}$) are assumed to be drawn from



$$K_{NC,i} \sim \text{Binomial}\big(\theta_{NC,i}, N_{NC,i}\big)$$
$$\theta_{NC,i} = \Phi\big(\phi_{NC,i}\big)$$
$$\phi_{NC,i} \sim \text{Normal}\big(\mu + \alpha/2, \sigma^2\big)$$

$$K_{AD,j} \sim \text{Binomial}\big(\theta_{AD,j}, N_{AD,j}\big)$$
$$\theta_{AD,j} = \Phi\big(\phi_{AD,j}\big)$$
$$\phi_{AD,j} \sim \text{Normal}\big(\mu - \alpha/2, \sigma^2\big)$$

$$\mu \sim \text{Normal}\big(0, 1\big)$$
$$\sigma \sim \text{Uniform}\big(0, 10\big)$$
$$\alpha = \delta \times \sigma$$

$$H_0 : \delta = 0$$
$$H_1 : \delta \neq 0$$
$$H_2 : \delta > 0$$

Figure 16.7: Bayesian graphical model for the Geurts data. In a between-subjects design, 26 typically developing children (i.e, "normal controls", NC) and 52 children with ADHD (AD) performed the Wisconsin Card Sorting Test.

group-level Normal distributions. Denoting the grand mean by μ, and the group difference in means by α, the group-level Normal distribution for the normal controls is defined as $N(\mu + \alpha/2, \sigma^2)$ and that for the ADHD children as $N(\mu - \alpha/2, \sigma^2)$, where σ denotes the standard deviation for the group-level distribution.

Figure 16.7 shows the model in graphical form. As in Figure 16.5, the hierarchical structure of the model is accommodated by plate notation, enclosing with square boundaries subsets of the graph that have independent replications. Because every child participants in only one of the two conditions, the design is "between-subjects" and the square boundaries enclose each condition separately.

For the parameters that are not subject to statistical test (i.e., μ and σ) we specified uninformative priors. The prior for the grant mean μ is a standard Normal, which on the rate scale translates to a uniform distribution from 0 to 1 (cf. Rouder & Lu, 2005, p. 588). For σ, we chose a prior that is uniform from 0 to 10. As in the previous example, the key aspect of our model is a parameter δ that quantifies effect size, $\delta = \alpha/\sigma$. We again use the "unit information" standard normal prior on δ, completing the specification of the model.

Hypothesis testing now proceeds as before. The null hypothesis states that normal controls and ADHD

children perform the same on the WCST, and hence the effect size is zero: $H_0 : \delta = 0$. The unrestricted alternative hypothesis states that there is a difference in performance, and hence $H_1 : \delta \neq 0$. Lastly, the order-restricted hypothesis states that normal controls perform better than ADHD children, such that $H_2 : \delta > 0$. These tests are hierarchical extensions of the Bayesian one-sample t-test; as in the previous example, the difference is that our hierarchical t-tests are defined on the level of individual parameters instead of raw data. Below we examine the unrestricted analysis (i.e., H_0 versus H_1) and the restricted analysis (i.e., H_0 versus H_2) in turn.

Unrestricted Analysis

We implemented our Bayesian hierarchical two-sample t-test by means of the Savage-Dickey procedure. The following code implements the graphical model in WinBUGS.

```
model
{
  for(i in 1:26) # 26 Normal Control Participants
  {
    # Binomial Distributions for Observed Counts:
    KNC[i] ~ dbin(thetaNC[i],NNC[i])
    # Transformation to Parameters on the Probit Scale:
    thetaNC[i] <- phi(phiNC[i])
    # Individual Parameters Come From a Group-Level Distribution:
    phiNC[i] ~ dnorm(muNC,tau)
    # NB. tau is the precision, defined as 1/variance
  }

  for(j in 1:52) # 52 ADHD Participants
  {
    KAD[j]      ~ dbin(thetaAD[j],NAD[j])
    thetaAD[j] <- phi(phiAD[j])
    phiAD[j]    ~ dnorm(muAD,tau)
  }

  muNC <- mu + (.5*alpha)
  muAD <- mu - (.5*alpha)
  # NB. mu is the grand mean, alpha is the effect (i.e., the group difference)

  # Group-Level Priors:
  mu ~ dnorm(0,1)
  # NB. This prior for mu corresponds to a uniform prior the rate scale,
  # ranging from 0 to 1.

  # Uninformative Prior on the Group-Level Standard Deviation:
  sigma ~ dunif(0,10)
  # Transformation from Standard Deviation to Precision:
  tau   <- pow(sigma,-2)
  alpha <- delta * sigma
  # NB. This allows one to put a prior on effect size delta (cf. Rouder et al., PBR)

  # The "Unit Information Prior" on Effect Size delta (cf. Rouder et al., PBR):
  delta ~ dnorm(0,1)
}
```

The code `Geurts_1.m` (Matlab) or `Geurts_1.R` (R) calls WinBUGS to draw samples from the posterior and the prior for the rate difference δ, and the left panel of Figure 16.8 visualizes the result. The ADHD

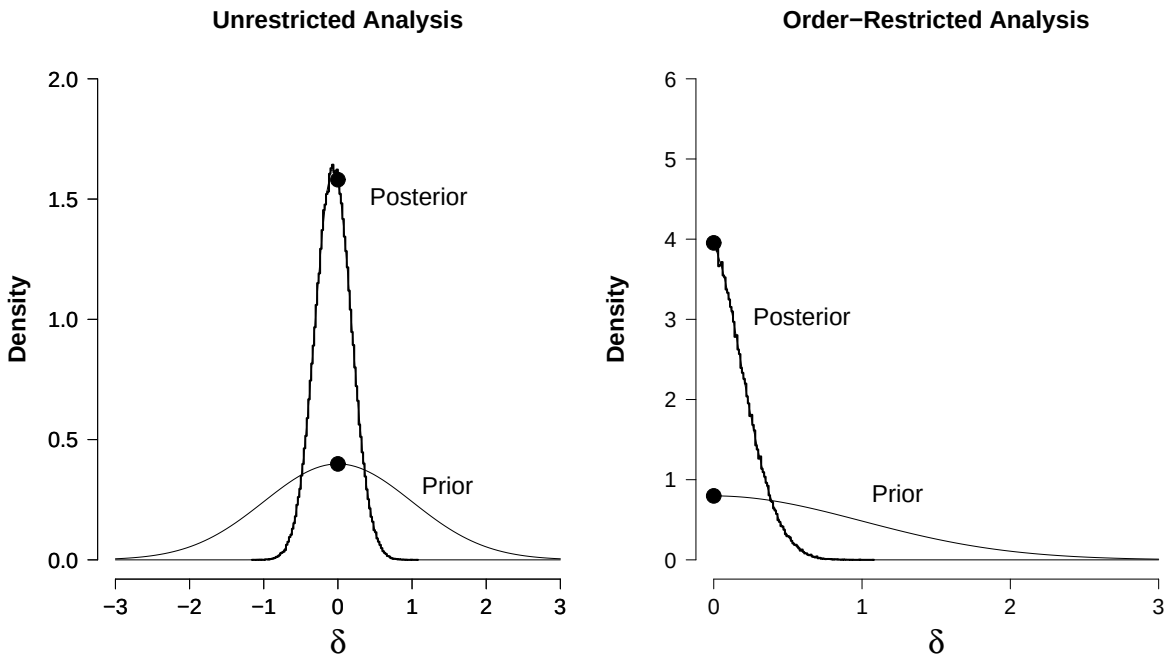


Figure 16.8: Prior and posterior distribution of the effect size δ for the hierarchical analysis of the Geurts data (left panel: unrestricted analysis; right panel: order-restricted analysis) For the prior, the thin line gives the analytical result; for the posterior, the thick line gives the histogram and the thin line gives the logspline density estimate. The dots indicate the height of the two distributions at $\delta = 0$.

children performed slightly better than the normal controls, and this is reflected in a posterior distribution for δ which is slightly asymmetrical around zero, assigning more mass to negative than to positive values of δ. The Bayesian 95% confidence interval for δ is $(-0.54, 0.42)$.

The left panel of Figure 16.8 also shows that the data have made the value $\delta = 0$ more likely than it was before (i.e., at $\delta = 0$, the posterior is higher than the prior). Specifically, the ratio of the heights yields $BF_{01} = 3.96$, which indicates that the data are about four times more likely under H_0 than they are under H_1. Thus, the data support the claim that normal controls and ADHD children perform equally well on the WCST over the claim that these groups perform differently.

Order-Restricted Analysis

The order-restricted hypothesis states that normal controls outperform children with ADHD on the WCST (i.e., $H_2 : \delta > 0$). This hypothesis may be entertained because it is plausible *a priori*; However, the data show that, if anything, the reverse is true: the mean percentage of correct card selections was 1.5% higher for the group of ADHD children than for the normal controls. What can we expect when we test $H_0 : \delta = 0$ versus $H_2 : \delta > 0$?

First, note that the posterior for δ is not far from being symmetrical around zero. If it were completely symmetrical, the height of both the prior and the posterior is multiplied by 2, so that their ratio stays the same. Second, the the posterior for δ is not quite symmetrical around zero, and assigns slightly more mass to values that are inconsistent with H_2. This will slightly increase the support for H_0 over H_2. These two considerations lead us to expect that the evidence in favor of H_0 over H_2 (i.e., BF_{02}) will be slightly larger than that of H_0 over H_1 (i.e., $BF_{01} = 3.96$)

The code `Geurts_2.m` (Matlab) or `Geurts_2.R` (R) calls WinBUGS to draw samples from the posterior and the prior for the rate difference δ, and the right panel of Figure 16.8 visualizes the result. A quantitative

comparison of the height of the prior and posterior at $\delta = 0$ confirms our expectation that the order-restriction slightly increases the evidence in favor of H_0. Specifically, the logspline estimate yields $BF_{02} = 4.94$. Thus, under H_0 the data are about five times more likely than they are under the order-restricted alternative, a result that is slightly more convincing than the one obtained when H_0 is pitted against the unrestricted alternative. In sum, the data support the assertion that normal controls and children with ADHD perform similarly on the WCST, even though the evidence is not overwhelming.

Exercises

1. For the unrestricted test ($H_0 : \delta = 0$ versus $H_2 : \delta \neq 0$), what is the maximum support in favor of H_0 that you could possibly observe, given the present number of subjects, and given that the average rate of correct card sorts is 65%?
2. What is the maximum support for the restricted test (i.e., $H_0 : \delta = 0$ versus $H_2 : \delta > 0$)?

Part IV

Advanced Topics

CHAPTER 17

GETTING STARTED WITH WBDEV

WITH RUUD WETZELS

17.1 What is WBDev?

For some psychological modeling applications, it is highly desirable to define one's own functions and distributions. In particular, user-defined functions and distributions greatly facilitate the use of psychological process models such as the Attention Learning Covering map (ALCOVE; Kruschke, 1992), the Generalized Context Model for category learning (GCM; Nosofsky, 1986), the Expectancy-Valence model for decision-making (Busemeyer & Stout, 2002), the SIMPLE model of memory (Brown et al., 2007), or the Ratcliff diffusion model of response times (Ratcliff, 1978).

The ability to implement these user–defined functions and distributions can be achieved through the use of the WinBUGS Development Interface (WBDev; Lunn, 2003), an add–on program that allows the user to hand–code functions and distributions in Component Pascal (e.g., `http://en.wikipedia.org/w/index.php?title=Component+Pascal`). The use of WBDev brings several advantages. For instance, complicated WBDev components lead to faster computation than their counterparts programmed in straight WinBUGS code. Moreover, some models will only work properly when implemented in WBDev. Another advantage of WBDev is that it compartmentalizes the code, resulting in scripts that are easier to understand, communicate, adjust, and debug. A final advantage of WBDev is that it allows the user to program functions and distributions that are simply not available in WinBUGS, but may be central components of psychological models (Donkin, Averell, Brown, & Heathcote, in press; Vandekerckhove, Tuerlinckx, & Lee, in press). In this chapter, we will demonstrate how to implement a new distribution in WBDev, using the shifted Wald distribution (Heathcote, 2004; Schwarz, 2001) as an example.

17.2 Installing WBDev

Before you can begin hard–coding your own functions and distributions, you need to download and install two programs; WBDev and BlackBox.[1] To make sure that all programs function properly, they have to be installed in the order given below.

1. Installing WinBUGS Development Interface (WBDev)

Download WBDev from `http://www.winbugs-development.org.uk/`. Unzip the executable (i.e., `WBDev.exe`) in your WinBUGS directory `./Program Files/WinBUGS14`. Then start WinBUGS, open the"`wbdev01_09_04.txt`" file and follow the instructions at the top of the file. During the process, WBDev will create its own directory `/WinBUGS14/WBDev`.

2. Installing BlackBox Component Builder

BlackBox is a development environment for programs written in Component Pascal and this includes WinBUGS. BlackBox can be downloaded from `http://www.oberon.ch/blackbox.html`. At the time of writing, the latest version is 1.5. Install BlackBox in the default directory: `./Program Files/BlackBox Component Builder 1.5`. Go to the WinBUGS directory and select all files (press "`Ctrl+A`") and copy them (press "`Ctrl+C`"). Next, open the BlackBox directory and paste the copied files in there (press "`Ctrl+V`"). Select " `Yes to all`" if asked about replacing files. Once this is done, you will be able to open BlackBox and run WinBUGS from inside BlackBox. This completes the installation of the software, and you can start to write our own functions and distributions.

[1] At the time of writing, all programs are available without charge.

17.3 Using WBDev: An Example Using the Shifted Wald Distribution

Statistical distributions are invaluable in psychological research. For example, in the simple rate problem discussed in Chapter 3, we used the binomial distribution to model our data. WinBUGS comes equipped with an array of predefined distributions, but it does not include some distributions that are potentially useful for psychological modeling. Using WBDev, researchers can augment WinBUGS to include these desired distributions, such as the shifted Wald distribution.

The Shifted Wald Distribution

Many psychological models use response times (RTs) to infer latent psychological properties and processes (Luce, 1986). One common distribution used to model RTs is the inverse Gaussian or Wald distribution (Wald, 1947). This distribution represents the density of the first passage times of a Wiener diffusion process toward a single absorbing boundary, as shown in Figure 17.1, using three parameters.

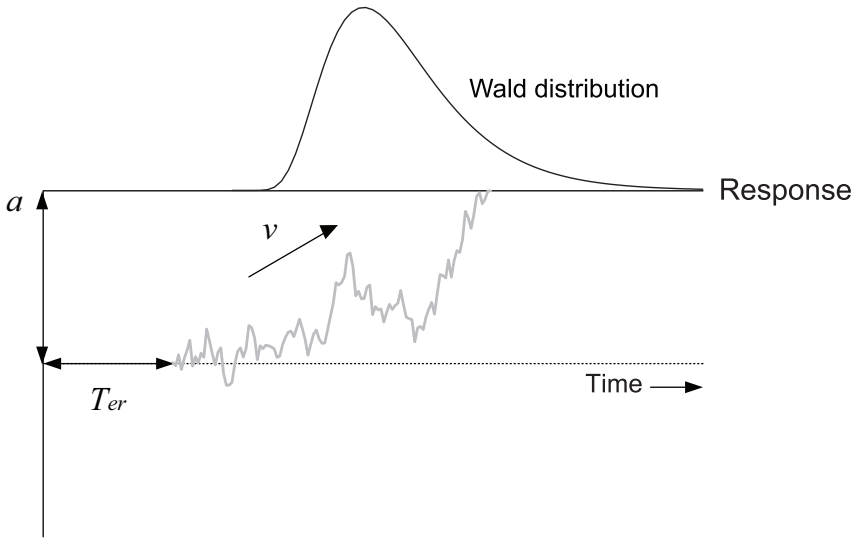


Figure 17.1: A diffusion process with one boundary. The shifted Wald parameter a reflects the separation between the starting point of the diffusion process and the absorbing barrier, v reflects the drift rate of the diffusion process and T_{er} is a positive–valued parameter that shifts the entire distribution.

The parameter v reflects the drift rate of the diffusion process. The parameter a reflects the separation between the starting point of the diffusion process and the absorbing barrier. The third parameter, T_{er}, is a positive–valued parameter that shifts the entire distribution. The probability density function for this shifted Wald distribution is given by:

$$f(t|v,a,T_{er}) = \frac{a}{\sqrt{2\pi(t-T_{er})^3}} \exp\left\{ -\frac{[a-v(t-T_{er})]^2}{2(t-T_{er})} \right\}, \tag{17.1}$$

which is unimodal and positively skewed. Because of these qualitative properties, it is a good candidate for fitting empirical RT distributions. As an illustration, Figure 17.2 shows changes in the shape of the shifted Wald distribution as a result of changes in the shifted Wald parameters v, a, and T_{er}.

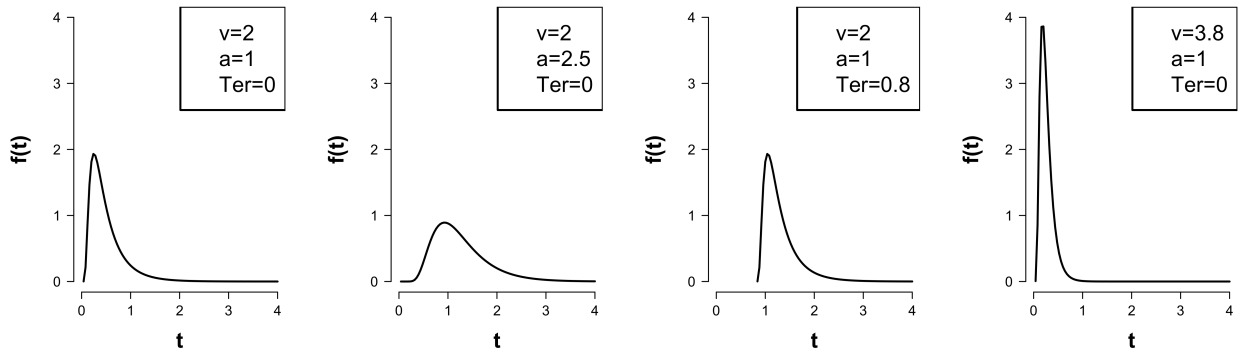


Figure 17.2: Changes in the shape of the shifted Wald distribution as a result of changes in the parameters v, a and T_{er}. Each panel shows the shifted Wald distribution with different combinations of parameters.

The shifted Wald parameters have a clear psychological interpretation (e.g., Heathcote, 2004; Luce, 1986; Schwarz, 2001, 2002). Participants are assumed to accumulate noisy information until a predefined threshold amount is reached and a response is initiated. Drift rate v quantifies task difficulty or subject ability, response criterion a quantifies response caution, and the shift parameter T_{er} quantifies the time needed for non-decision processes (Matzke & Wagenmakers, in press). Experimental paradigms in psychology for which it is likely that there is only a single absorbing boundary include saccadic eye movement tasks with few errors (Carpenter & Williams, 1995), go/no–go tasks (Gomez, Ratcliff, & Perea, 2007) or simple reaction time tasks (Luce, 1986, pp. 51–57).

The WBDev Script

The WBDev script for implementing the shifted Wald distribution is available in the `ShiftedWald.txt` file. In this section, we show only some crucial parts of the WBDev script. The numbers (*X*) correspond to the numbers in the `ShiftedWald.txt` WBDev script.

(*1*) `MODULE WBDevShiftedWald;`

The name of the module is typed here. We want to name our module ShiftedWald. The name of the module (so the part after MODULE WBDev...) has to start with a capital letter.

(*2*) `drift = 0; bound = 1; shift = 2;`

The parameters of the distribution, which, in this case are the drift rate v, response caution a and shift T_{er}.

(*3*)
```
log2Pi:  REAL;
fact-:  WBDevUnivariate.Factory;
```

Here global variables can be declared. A global variable is loaded only once, but the value of the variable is usually needed many times.

(*4*) `args := "sss";`

We have to declare what type of arguments are the input of the distribution. In this case these are the three scalar parameters of the shifted Wald distribution.

(*5*)
```
isDiscrete := FALSE;
canIntegrate := FALSE;
```

The first line describes whether samples from the distribution are discrete or continuous. When the distribution is discrete, `isDiscrete` should be set to "TRUE". When the distribution is continuous, it should be set to "FALSE". For the shifted–Wald distribution `isDiscrete` is "FALSE".

The second line defines whether the cumulative distribution is to be provided. If so, `canInteg- rate` should be set to "TRUE". If this is set to true, an algorithm should be provided at **(*11*)**. We set `canIntegrate` to "FALSE" because we did not implement the cumulative distribution.

(*6*) `lower := Ter;`
`upper := INF;`

This part of the code should define the natural bounds of the distribution. In our case, we take T_{er} as a lower bound and INF (meaning $+\infty$) as an upper bound.

(*7*) `x := t-Ter;`
`value:= -0.5*log2Pi + Math.Ln(a) - 1.5*Math.Ln(x) - ((a-v*x)*(a-v*x))/(2*x);`

As the name implies, this is the part where the full log likelihood of the distribution is defined.

(*8*) `LogFullLikelihood(node, value);`

Sometimes WinBUGS can ignore the normalizing constants. When that is the case, WinBUGS calls `LogPropLikelihood(.)`. In our example, we refer back to the full log likelihood function.

(*9*) `LogFullLikelihood(node, value);`

Occasionally, WinBUGS can make use of the `Logprior(.)` procedure, which is proportional to the real log–prior function. In other words, this procedure omits the additive constants on the log scale. In our example, we just refer back to the full log likelihood function.

(*10*) Here the cumulative distribution can be defined in case `canIntegrate` at **(*7*)** had been set to TRUE. Because we have set `canIntegrate` to FALSE, we do not define anything in this section.

(*11*) The `DrawSample(.)` procedure returns a pseudo–random number from the new distribution. We do not use this function, because we do not need to draw values from the new distribution. You would have to do this when you have missing values.

(*12*) `END WBDevShiftedWald.`

The last thing that needs to be done is to make sure that the name of the module at the end is the same as the name at the top of the file. The last line has to end with a period.

Open BlackBox, and copy the content of the `ShiftedWald.txt` file to a new file. You need to compile the function by pressing `Ctrl+K`. Syntax errors cause WBDev to return an error message. Save this file as `ShiftedWald.odc` and copy it into the appropriate BlackBox directory, `.../BlackBox Component Builder 1.5/WBdev/Mod`.

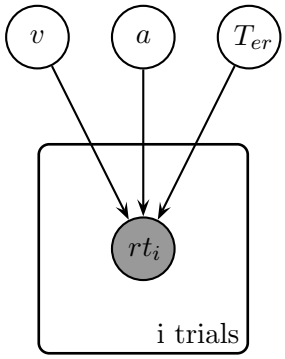
We are, however, not entirely ready yo use the shifted Wald distribution yet. WBDev needs to know that there exists a distribution called ShiftedWald; WBDev also needs to know what the input looks like (i.e., how many inputs are there, what order are they presented, and are they scalars or vectors?) To accomplish this, open the distribution file `distributions.odc` in the directory `.../BlackBox Component Builder 1.5/WBdev/Rsrc`. Add the line: `s ~ "ShiftedWald"(s,s,s) "WBDevShiftedWald.Install"` and then save it. The next time you start BlackBox, the program will know that there exists a distribution called ShiftedWald, and that the inputs are three scalars (single numbers).

Application of the Shifted Wald Distribution

We will illustrate the application of the shifted Wald distribution using a real dataset from a lexical decision task (Wagenmakers, Ratcliff, Gomez, & McKoon, 2008). Nineteen participants had to quickly decide whether a visually presented letter string was a word (e.g., table) or a nonword (e.g., drapa). The graphical model representation for this problem is shown in Figure 17.3.

The following code implements the graphical model in WinBUGS.

```
model {
   # The shifted Wald parameters are drawn from their prior distributions
   v ~ dunif(0, 10)
   a ~ dunif(0, 10)
```



$$v \sim \text{Uniform}(0, 10)$$
$$a \sim \text{Uniform}(0, 10)$$
$$T_{er} \sim \text{Uniform}(0, 1)$$

$$rt_i \sim \text{ShiftedWald}(v, a, T_{er})$$

Figure 17.3: Graphical model for the shifted Wald analysis.

```
   Ter ~ dunif(0, 1)

   # The data are shifted Wald distributed
   for (i in 1:nrt) {
      rt[i] ~ ShiftedWald(v,a,Ter)
   }
}
```

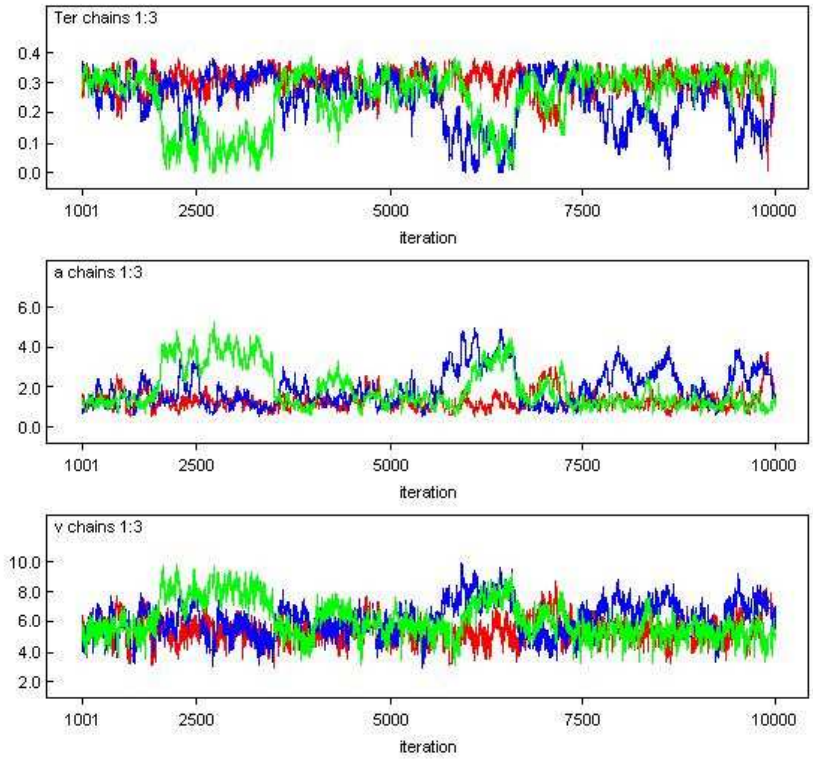


Figure 17.4: The MCMC chains of the marginal posteriors of all three individual shifted Wald parameters, v, a and Ter.

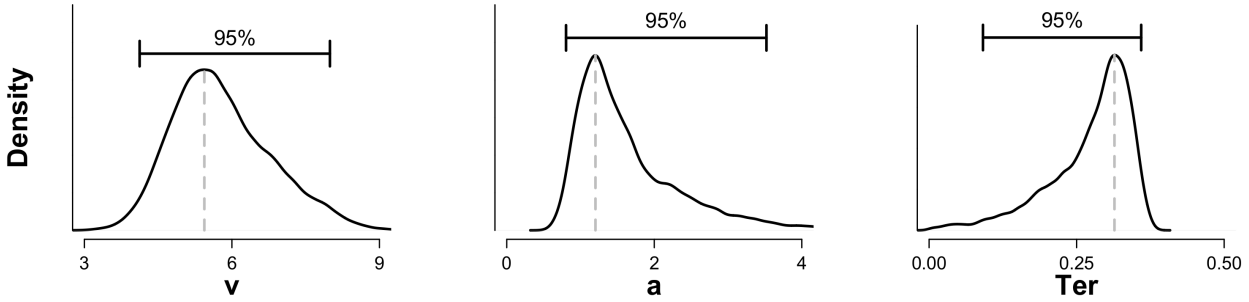


Figure 17.5: The posterior distributions of the three shifted Wald parameters v, a and T_{er}. The dashed gray lines indicate the modes of the posterior distributions at $v = 5.57$, $a = 1.09$ and $T_{er} = .33$. The 95% confidence intervals for v, a and T_{er} extend from 4.12 to 8.00, from .80 to 3.52 and from .09 to .36, respectively.

The priors for v and a are uniform distributions that range from 0 to 10 (i.e., `v` $\sim$ `dunif(0,10)` and `a` $\sim$ `dunif(0,10)`). The prior for T_{er} is a uniform distribution that ranges from 0 to 1 (i.e., `Ter` $\sim$ `dunif(0,1)`. With the priors in place, we can use our ShiftedWald function to estimate the posterior distributions for the three model parameters v, a and T_{er} (i.e., `rt[i]` $\sim$ `ShiftedWald(v,a,Ter)`).

The code `ShiftedWald_1.m` (Matlab, requires the `Statistics Toolbox`) or `ShiftedWald_1.R` (R) loads the RTs of correct "word" responses of the first participant, and then calls WinBUGS to fit the RTs to the shifted Wald distribution. Note that `bugs.dir` in the Matlab code and `bugsdir` in the R code must be set to the location of your copy of BlackBox and not to the location of the WinBUGS software.

After you run the code, WinBUGS should show MCMC chains similar to those shown in Figure 17.4. The chains do not look like fat hairy caterpillars. They seem to have a lot of freedom to move around the parameter space, so we cannot be certain that the chains have converged properly. To assess convergence more formally, we ran three chains using different starting points for each chain. Next, we calculated Rhat to check whether the chains have converged to the same stationary distribution. For each parameter, Rhat is smaller than 1.1, so we can tentatively assume that the chains have converged.

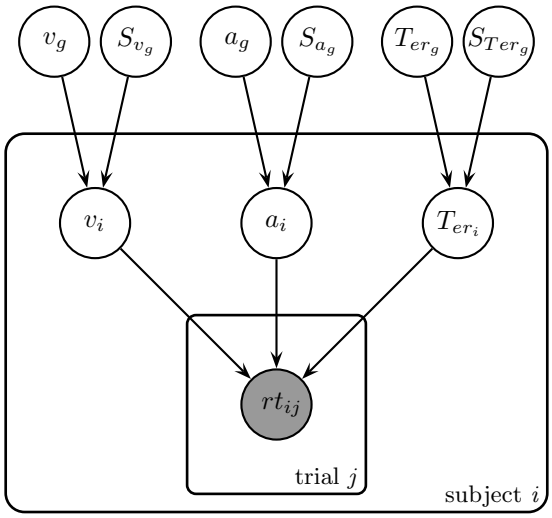
Figure 17.5 shows the posterior distributions of the three shifted Wald parameters, v, a and T_{er}. One thing that stands out is that the posterior distributions of the shifted Wald parameters are very spread out across the parameter space. The 95% confidence intervals for v, a and T_{er} extend from 4.12 to 8.00, from .80 to 3.52 and from .09 to .36, respectively. It seems that data from only one participant are not enough to yield very accurate estimates of the shifted Wald parameters. In the following section we show how our estimates will improve when we use a hierarchical model and analyze all participants simultaneously.

Application of the Shifted Wald Distribution: A Hierarchical Extension

In an experimental setting, the problem of few data per participant can be addressed by hierarchical modeling (Farrell & Ludwig, 2008; Gelman & Hill, 2007; Rouder, Sun, Speckman, Lu, & Zhou, 2003; Shiffrin, Lee, Kim, & Wagenmakers, 2008). In our shifted Wald example, each subject is assumed to generate their data according to the shifted Wald distribution, but with different parameter values. We extend the individual analysis and assume that the parameters for each subject are chosen from a normal distribution. This means that all individual participants are assumed to have their shifted Wald parameters drawn from the same group distribution, allowing all the data provided by all the participants to be used for inferring parameter values, without making the unrealistic assumption that participants are identical copies of each other. The graphical model representation for this problem is shown in Figure 17.6

The model file that implements the hierarchical shifted Wald analysis is shown below.

```
model {
   # prior distributions for group means:
```



$$v_g \sim \text{Uniform}(0, 10)$$
$$S_{v_g} \sim \text{Uniform}(0, 5)$$
$$a_g \sim \text{Uniform}(0, 10)$$
$$S_{a_g} \sim \text{Uniform}(0, 5)$$
$$T_{er_g} \sim \text{Uniform}(0, 1)$$
$$S_{Ter_g} \sim \text{Uniform}(0, 1)$$
$$v_i \sim \text{Gaussian}(v_g, \tfrac{1}{S^2_{v_g}})$$
$$a_i \sim \text{Gaussian}(a_g, \tfrac{1}{S^2_{a_g}})$$
$$T_{er_i} \sim \text{Gaussian}(T_{er_g}, \tfrac{1}{S^2_{Ter_g}})$$
$$rt_{ij} \sim \text{shiftedWald}(v_i, a_i, T_{er_i})$$

Figure 17.6: Graphical model for the hierarchical shifted Wald analysis.

```
vg ~ dunif(0, 10)
ag ~ dunif(0, 10)
Terg ~ dunif(0, 1)

# prior distributions for group standard deviations:
sdvg ~ dunif(0,5)
sdag ~ dunif(0,5)
sdTerg ~ dbeta(1,1)

#transformation from group standard deviations to group precisions
#(i.e., 1/var, which is what WinBUGS expects
#as input to the dnorm distribution):

lambdavg <- pow(sdvg,-2)
lambdaag <- pow(sdag,-2)
lambdaTerg <- pow(sdTerg,-2)

# Data Come From a Shifted Wald
for (i in 1:ns) {    #subject loop
   #individual parameters drawn from group level
   #normals censored to be positive using the
   #I(0,) command:

   vi[i] ~ dnorm(vg, lambdavg)I(0,)
   ai[i] ~ dnorm(ag, lambdaag)I(0,)
   Teri[i] ~ dnorm(Terg,lambdaTerg)I(0,)

   #The data are shifted Wald distributed
   for (j in 1:nrt[i]) {
      rt[i,j] ~ ShiftedWald(vi[i],ai[i],Teri[i])
   }
```

```
   }
}
```

The hierarchical analysis of the reaction time data proceeds as follows. The prior of the group means is a uniform distribution, ranging from 0 to 10 (i.e., `vg` $\sim$ `dunif(0,10)` and `ag` $\sim$ `dunif(0,10)`) or from 0 to 1 (i.e., `Terg` $\sim$ `dunif(0,1)`). The standard deviations are drawn from a uniform distribution ranging from 0 to 5 (i.e., `sdvg` $\sim$ `dunif(0,5)` and `sdag` $\sim$ `dunif(0,5)`) or from 0 to 1 (i.e., `sdTerg` $\sim$ `dunif(0,1)`). Next, the standard deviations have to be transformed to precisions (i.e., `lambdavg <- pow(sdvg,-2)`, `lambdaag <- pow(sdag, -2)` and `lambdaTerg <- pow(sdTerg,-2)`). Then, the individual parameters v_i, a_i and Ter_i are drawn from normal distributions with the corresponding group means and group precisions (i.e., `vi[i]` $\sim$ `dnorm(vg,lambdavg)I(0,)`, `ai[i]` $\sim$ `dnorm(ag, lambdaag)I(0,)` and `Teri[i]` $\sim$ `dnorm(Terg, lambdaTerg)I(0,)`). For each individual, the data are distributed according to a shifted Wald distribution with their own individual parameters.

The code `ShiftedWald_2.m` (Matlab, requires the `Statistics Toolbox`) or `ShiftedWald_2.R` (R) loads the RTs of correct "word" responses of the nineteen participants, and then calls WinBUGS to fit the RTs to the shifted Wald distribution. We first focus on the group mean parameters v_g, a_g and T_{er_g}. Figure 17.7 shows the MCMC chains for the three shifted Wald parameters. To check for convergence, we ran three chains, with all three having a different starting position, and then calculated Rhat. The chains appear to have converged, an impression that is supported by Rhat values close to 1 (Rhat for Ter_g, a_g and v_g is approximately 1.).

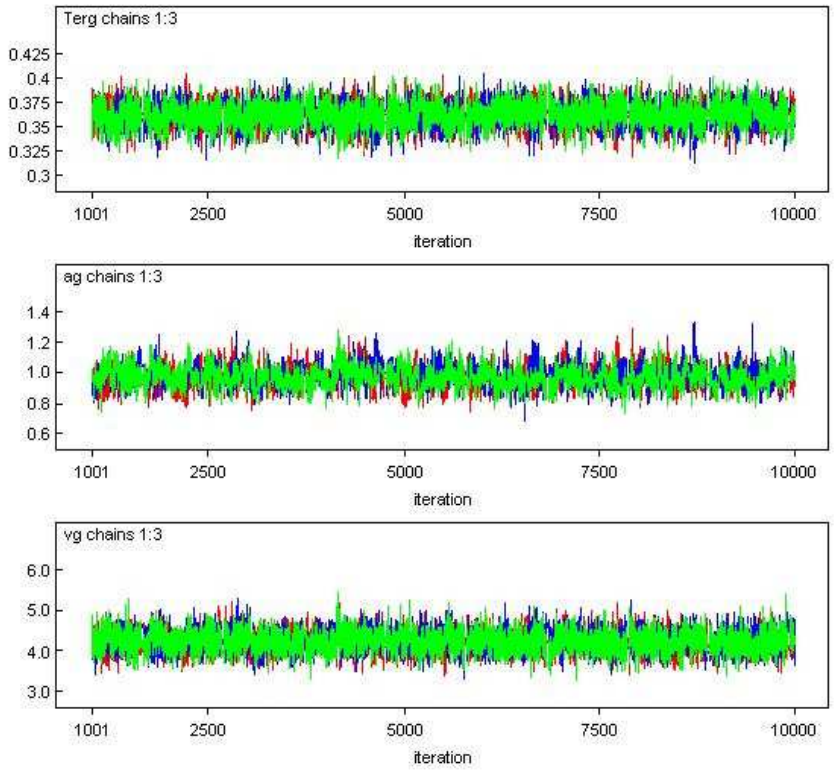


Figure 17.7: Three chains, consisting of 9000 MCMC draws each, from the posterior distributions of the three "group–level" shifted Wald parameters, v_g, a_g and T_{er_g}.

Figure 17.8 shows the posterior distributions of the shifted Wald group–mean parameters. The distributions indicate that there is relatively little uncertainty about the parameter values. The posterior distributions of the group–mean parameters are concentrated around their modes $v_g = 4.27$, $a_g = .97$ and

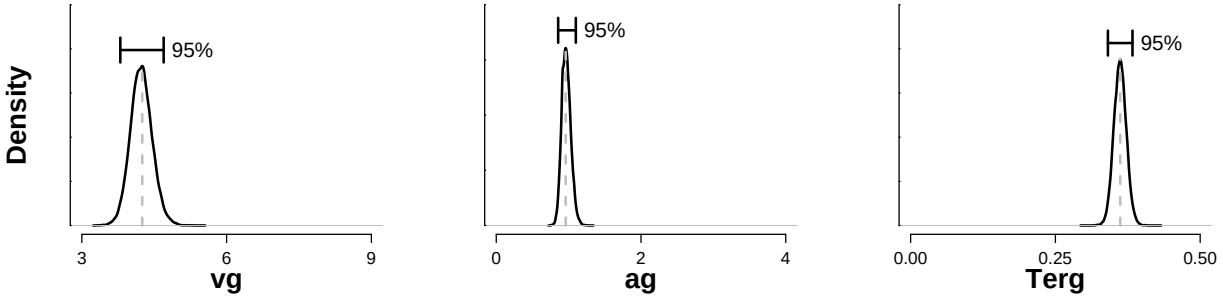


Figure 17.8: The posterior distribution of the three "group–level" shifted Wald parameters v_g, a_g and T_{er_g}. The dashed gray lines indicate the modes of the posterior distributions at $v_g = 4.27$, $a_g = .97$ and $T_{er_g} = .36$. The 95% confidence intervals for v_g, a_g and T_{er_g} extend from 3.80 to 4.70, from .85 to 1.10 and from .34 to .38, respectively.

$T_{er_g} = .36$. The 95% confidence intervals for v_g, a_g and T_{er_g} extend from 3.80 to 4.70, from .85 to 1.10 and from .34 to .38, respectively.

It is informative to consider the influence of the hierarchical extension on the individual estimates for the shifted Wald parameters. Specifically, we can examine the MCMC chains for the same subject that we analyzed in the individual shifted Wald analysis, but now in the hierarchical setting.

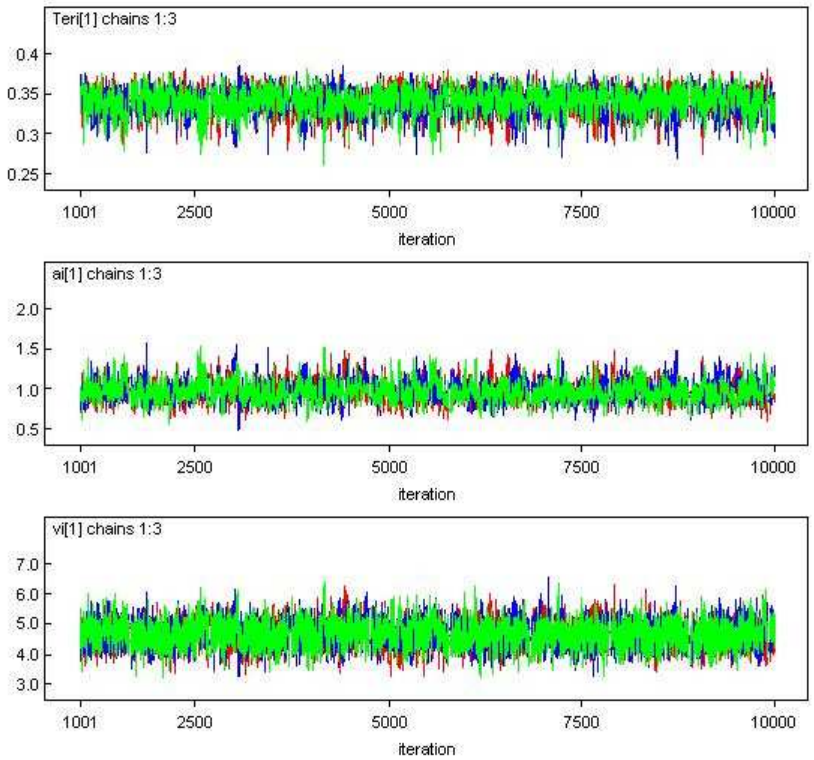


Figure 17.9: The MCMC chains of the marginal posteriors of all three individual shifted Wald parameters, v, a and Ter, analyzed using a hierarchical model.

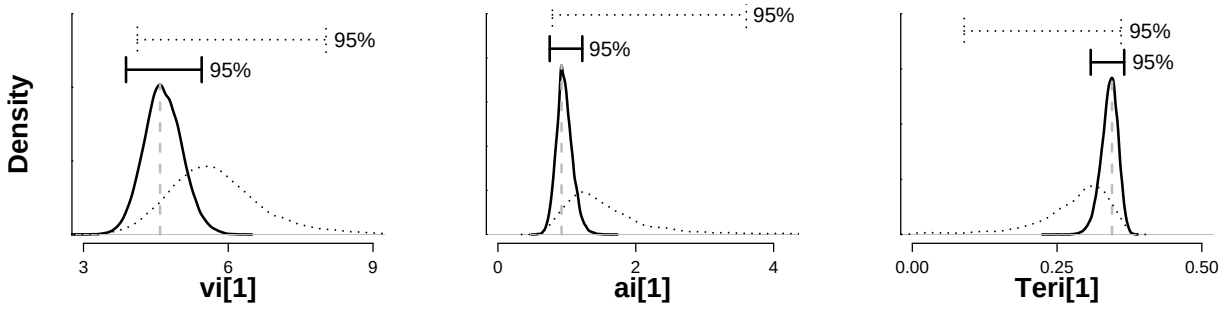


Figure 17.10: The posterior distribution of the three individual shifted Wald parameters v_i, a_i and T_{er_i} from the hierarchical analysis (solid lines) and the individual analysis (dotted lines). The dashed gray lines indicate the modes of the posterior distributions from the hierarchical analysis at $v_i[1] = 4.27$, $a_i[1] = .97$ and $T_{er_i}[1] = .36$. The 95% confidence intervals in the hierarchical model for $v_i[1]$, $a_i[1]$ and $T_{er_i}[1]$ extend from 3.86 to 5.49, from 0.75 to 1.24 and from .31 to .37,respectively.

After you run the `ShiftedWald_2.m` (Matlab) or `ShiftedWald_2.R` (R) script for the hierarchical analysis of the shifted Wald example, WinBUGS should show three MCMC chains similar to the ones shown in Figure 17.9. The chains are better behaved than the chains from the individual analysis (Figure 17.4). The hierarchical extension leads to a practical improvement, through faster convergence for the computational MCMC estimation process. However, the hierarchical extension also leads to a theoretical improvement because compared to the individual analysis, the chains appear much less diffuse. This shows that the hierarchical model leads to a better understanding of the model parameters.

To underscore this point, Figure 17.10 shows the posterior distributions of the individual shifted Wald parameters, for both the hierarchical analysis and the individual analysis. It is clear that the posterior distributions of the shifted Wald parameters are less spread out in the hierarchical analysis than in the individual analysis. Also, the parameter estimates from the hierarchical analysis are slightly different than those from the individual analysis. More precisely, they seem to have moved towards their common group mean. This effect is called *shrinkage*, and is a standard and important property of hierarchical models (Gelman, Carlin, Stern, & Rubin, 2004).

In sum, the WBDev implementation of the shifted Wald distribution enables researchers to use WinBUGS to infer shifted Wald parameters from reaction time data. Not only does WinBUGS allow straightforward analyses on individual data, it also makes it easy to add hierarchical structure to the model. This can greatly improve the quality of the posterior estimates, and is often a very sensible and informative way of analyzing data.

17.4 Online Help and Useful URLs

Further information on WBDev, including various examples of WBDev scripts, is available at `http://www.ruudwetzels.com/index.php?src=WBDev`.

BIBLIOGRAPHY

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

Aitchison, J., & Dunsmore, I. R. (1975). *Statistical prediction analysis*. Cambridge: Cambridge University Press.

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*(1), 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.

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.

Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory*. New York: Wiley.

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.

Brooks, S. P., & Gelman, A. (1997). 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*(1), 539–576.

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

Busemeyer, J. R., & Stout, J. C. (2002). A contribution of cognitive decision models to clinical assessment: Decomposing performance on the Bechara gambling task. *Psychological Assessment*, *14*, 253–262.

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.

Carlson, B. W., & Yates, J. F. (1989). Disjunction errors in qualitative likelihood judgment. *Organizational Behavior and Human Decision Processes*, *44*, 368–379.

Carpenter, R. H. S., & Williams, M. L. L. (1995). Neural computation of log likelihood in control of saccadic eye movements. *Nature*, *377*, 59–62.

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

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.

Dennis, S., & 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.

Donkin, C., Averell, L., Brown, S., & Heathcote, A. (in press). Getting more from accuracy and response time data: Methods for fitting the linear ballistic accumulator model. *Behavior Research Methods*.

Draper, D. (1995). Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society, Series B*, *57*, 45–97.

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

Fantino, E., Kulik, J., Stolarz–Fantino, S., & Wright, W. (1997). The conjunction fallacy: A test of averaging hypotheses. *Psychonomic Bulletin & Review*, *4*, 96–101.

Farrell, S., & Ludwig, C. (2008). Bayesian and maximum likelihood estimation of hierarchical response time models. *Psychonomic bulletin & review*, *15*, 1209.

Fleiss, J. L., Levin, B., & Paik, M. C. (2003). *Statistical methods for rates and proportions* (Third ed.). New York: 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.

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

Garner, W. R. (1974). *The Processing of Information and Structure*. Potomac, MD: Erlbaum.

Gati, I., & Tversky, A. (1982). Representations of qualitative and quantitative dimensions. *Journal of Experimental Psychology: Human Perception and Performance*, *8*(2), 325–340.

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*(3), 515–534.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis (2nd ed.)*. Boca Raton (FL): Chapman & Hall/CRC.

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., & Goldstein, D. G. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, *103*(4), 650–669.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (Eds.). (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.

Gomez, P., Ratcliff, R., & Perea, M. (2007). A model of the go/no–go task. *Journal of Experimental Psychology: General*, *136*, 389-413.

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

Good, I. J. (1985). Weight of evidence: A brief survey. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics 2* (pp. 249–269). New York: Elsevier.

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: 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.

Heathcote, A. (2004). Fitting Wald and ex–Wald distributions to response time data: An example using functions for the S–PLUS package. *Behavior Research Methods, Instruments, & Computers*, *36*, 678–694.

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 ScienceSociety* (pp. 923–928). Mahwah, NJ: Erlbaum.

Helm, C. E. (1959). *A Multidimensional Ratio Scaling Analysis of Color Relations*. (Princeton, NJ: Princeton University and Educational Testing Service)

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, *14*, 382-417.

Hoijtink, H., Klugkist, I., & Boelen, P. (2008). *Bayesian evaluation of informative hypotheses that are of practical value for social scientists*. New York: Springer.

Hsiao, C. K. (1997). Approximate Bayes factors when a mode occurs on the boundary. *Journal of the American Statistical Association*, *92*, 656–663.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge, UK: Cambridge University Press.

Jeffreys, H. (1961). *Theory of probability*. Oxford, UK: Oxford University Press.

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

Kass, R. E., & Vaidyanathan, S. K. (1992). Approximate Bayes factors and orthogonal parameters, with application to testing equality of two binomial proportions. *Journal of the Royal Statistical Society, Series B*, *54*, 129–144.

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.

Klugkist, I., Laudy, O., & Hoijtink, H. (2005). Inequality constrained analysis of variance: A Bayesian approach. *Psychological Methods*, *10*, 477–493.

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 volume 1: Statistical methods in clinical studies.* New York: Wiley.

Kruschke, J. K. (1992). ALCOVE: An exemplar–based connectionist model of category learning. *Psychological Review*, *99*, 22–44.

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

Lee, M. D. (2008). Three case studies in the Bayesian analysis of cognitive models. *Psychonomic Bulletin & Review*, *15*(1), 1–15.

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

Lindley, D. V. (1972). *Bayesian statistics, a review.* Philadelphia (PA): SIAM.

Liu, C. C., & Aitkin, M. (in press). Bayes factors: Prior sensitivity and model generalizability. *Journal of Mathematical Psychology.*

Liu, J., & Wu, Y. (1999). Parameter Expansion for Data Augmentation. *Journal of the American Statistical Association*, *94*(448).

Luce, R. D. (1986). *Response times.* New York: Oxford University Press.

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

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: Cambridge University Press.

MacMillan, N., & Creelman, C. D. (2004). *Detection theory: A user's guide (2nd ed.).* Hillsdale, NJ: Erlbaum.

Madigan, D., & Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, *89*, 1535–1546.

Martignon, L., & Laskey, K. B. (1999). Bayesian benchmarks for fast and frugal heuristics. In G. Gigerenzer & P. M. Todd (Eds.), *Simple heuristics that make us smart* (pp. 169–188). New York: Oxford University Press.

Mather, I., & Mather, C. (1979). The differential diagnosis and treatment of mental afflictions. *The Journal of Irreproducible Results*, *25*(4), 15–16.

Matzke, D., & Wagenmakers, E.-J. (in press). Psychological interpretation of exgaussian and shifted wald parameters: A diffusion model analysis. *Psychonomic Bulletin & Review.*

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., Forster, M. R., & Browne, M. W. (2000). Model selection [Special issue]. *Journal of Mathematical Psychology*, *44*(1–2).

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

Nilsson, H., Winman, A., Juslin, P., & Hansson, G. (2009). Linda is not a bearded lady: Configural weighting and adding as the cause of extension errors. *Manuscript submitted for publication.*

Nosofsky, R. (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: Wiley.

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

O'Hagan, A., & Forster, J. (2004). *Kendall's advanced theory of statistics vol. 2B: Bayesian inference (2nd ed.)*. London: Arnold.

Parsons, L. M., & Osherson, D. (2001). New evidence for distinct right and left brain systems for deductive andprobabilistic reasoning. *Cerebral Cortex*, *11*, 9545–965.

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for MCMC. *R News*, *6*, 7–11.

Poehling, K. A., Griffin, M. R., & Dittus, R. S. (2002). Bedside diagnosis of influenzavirus infections in hospitalized children. *Pediatrics*, *110*(1), 83–88.

Raftery, A. E. (1995). Bayesian model selection in social research. In P. V. Marsden (Ed.), *Sociological methodology* (pp. 111–196). Cambridge: Blackwells.

Ratcliff, R. (1978). A theory of memory retrieval. *Psychological Review*, *85*, 59–108.

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

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

Rouder, J. N., & Lu, J. (2005). An introduction to Bayesian hierarchical models with an application in the theory of signal detection. *Psychonomic Bulletin & Review*, *12*, 573–604.

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.

Rouder, J. N., Sun, D., Speckman, P., Lu, J., & Zhou, D. (2003). A hierarchical Bayesian statistical framework for response time distributions. *Psychometrika*, *68*, 589–606.

Rubin, D. C., Hinton, S., & Wenzel, A. (1999). The precise time course of retention. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *25*(5), 1161–1176.

Rubin, D. C., & Wenzel, A. E. (1996). One hundred years of forgetting: A quantitative description of retention. *Psychological Review*, *103*(4), 734–760.

Schwarz, W. (2001). The ex–Wald distribution as a descriptive model of response times. *Behavior Research Methods, Instruments, & Computers*, *33*, 457–469.

Schwarz, W. (2002). On the convolution of Inverse Gaussian and exponential random variables. *Communications in statistics. Theory and methods*, *31*, 2113–2121.

Shepard, R. N. (1980). Multidimensional scaling, tree-fitting, and clustering. *Science*, *210*, 390–398.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science*, *237*, 1317–1323.

Shepard, R. N. (1991). Integrality versus separability of stimulus dimensions: From an early convergenceof evidence to a proposed theoretical basis. In J. R. Pomerantz & G. L. Lockhead (Eds.), *The Perception of Structure: Essays in Honor of Wendell R. Garner* (pp. 53–71). Washington, DC: American Psychological Association.

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

Shiffrin, R., Lee, M., Kim, W., & Wagenmakers, E. (2008). A Survey of Model Evaluation Approaches with a Tutorial on Hierarchical Bayesian Methods. *Cognitive Science*, *32*, 37.

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 B*, *42*, 213–220.

Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2003). *WinBUGS user manual 1.4*. Cambridge, UK: MRC Biostatistic Unit.

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.

Todd, P. M., & Dieckmann, A. (2005). Heuristics for ordering cue search in decision making. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in Neural Information Processing Systems* (Vol. 17, pp. 1393–1400). Cambridge, MA: MIT Press.

Treat, T. A., MacKay, D. B., & Nosofsky, R. M. (1999). *Probabilistic Scaling: Basic Research and Clinical Applications.* Paper presented at the meeting of the Society for Mathematical Psychology,Santa Cruz.

Tversky, A., & Kahneman, D. (1983). Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review*, *90*, 293–315.

Uebersax, J. S. (1987). Diversity of decision-making models and the measurement of interrater agreement. *Psychological Bulletin*, *101*(1), 140–146.

Vandekerckhove, J., Tuerlinckx, F., & Lee, M. D. (in press). A Bayesian approach to diffusion process models of decision–making. *Proceedings of the 30th Annual Conference of the Cognitive Science Society*.

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., Ratcliff, R., Gomez, P., & Iverson, G. J. (2004). Assessing model mimicry using the parametric bootstrap. *Journal of Mathematical Psychology*, *48*, 28–50.

Wagenmakers, E.-J., Ratcliff, R., Gomez, P., & McKoon, G. (2008). A diffusion model account of criterion shifts in the lexical decision task. *Journal of Memory and Language*, *58*, 140–159.

Wagenmakers, E.-J., & Waldorp, L. (2006). Model selection: Theoretical developments and applications [Special issue]. *Journal of Mathematical Psychology*, *50*(2).

Wald, A. (1947). *Sequential analysis*. Wiley New York.

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

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.