CHAPTER 25

Tools in the Trunk

Contents

She changes her hair, and he changes his style,
She paints on her face, and he wears a fake smile,
She shrink wraps her head, and he stretches the truth,
But they'll always be stuck with their done wasted youth.[1]

This chapter includes some important topics that apply to many different models throughout the book. Please see the list of contents, above. The sections can be read independently of each other and at any time.

25.1. REPORTING A BAYESIAN ANALYSIS

Bayesian data analyses are not yet standard procedure in many fields of research, and no conventional format for reporting them has been established. Therefore, the researcher who reports a Bayesian analysis must be sensitive to the background knowledge of his or her specific audience, and must frame the description accordingly.

[1] One of the topics in this chapter is reparameterization, in which parameters of a model are transformed into new parameters. For example, a rectangle can be described in terms of its length and width, or in terms of its area and aspect ratio, where area = length × width and aspect ratio = length/width. Either way, it's the same rectangle. The poem personifies reparameterization.

Doing Bayesian Data Analysis, 2nd Edition
http://dx.doi.org/10.1016/B978-0-12-405888-0.00025-8

I once assigned an exercise to students in which they had to write up the results of a Bayesian analysis as it would appear in a research journal. Because I am a psychologist, a student then earnestly asked me, "Do we have to write the report as if it were for a "psychology" journal or for a "science" journal?" After swallowing my feeling of injury at the implication that psychology is not science, I politely asked the student to clarify the question. The student said, "For a psychology journal you have to explain what you did, but for a science journal the reader has to puzzle it out." I hope that all science journals will allow you to explain what you did.

Thinking about how to clearly report a Bayesian analysis also gets you to think clearly about the analysis itself. One recent article in a statistics education journal described a classroom exercise in which students had to generate topics to include in a report (Pullenayegum, Guo, & Hopkins, 2012, with discussion on my blog at http://doingbayesiandataanalysis.blogspot.com/2012/05/how-to-report-bayesian-analysis.html). The primary focus of the exercise was the conceptual clarification it encouraged in the students, not the resulting list. Nevertheless, it is interesting that the list generated by the students missed some points that I had indicated were essential in the 1st edition of this book, and which appear again below. I hope that the motivation for the points I offer below helps you to produce clear reports and to better understand Bayesian analysis.

25.1.1. Essential points

Recall the basic steps of a Bayesian analysis from Section 2.3 (p. 25): Identify the data, define a descriptive model, specify a prior, compute the posterior distribution, interpret the posterior distribution, and, check that the model is a reasonable description of the data. Those steps are in logical order, with each step building on the previous step. That logical order should be preserved in the report of the analysis. The essential points listed below mirror the basic mechanical steps of Bayesian analysis, preceded by an important preliminary step that might recede in importance as Bayesian methods become standard procedure.

- Motivate the use of Bayesian (non-NHST) analysis: Many audiences, including journal editors and reviewers, are comfortable with NHST and are not familiar with Bayesian methods. I have found that most editors and reviewers are eager to publish research that uses the best possible methods, including Bayesian analyses, but the editors and reviewers appreciate an explanation of why you have used Bayesian analysis instead of NHST. You may motivate your use of Bayesian data analysis on several grounds, depending in part on the particular application and audience. For example, Bayesian models are designed to be appropriate to the data structure, without having to make approximation assumptions typical in NHST (e.g., homogeneity of variances across groups, and normally distributed noise). The

inferences from a Bayesian analysis are richer and more informative than NHST because the posterior distribution reveals joint probabilities of combinations of parameter values. And, of course, there is no reliance on sampling distributions and *p* values to interpret the parameter estimates.

- Clearly describe the data structure, the model, and the model's parameters: Ultimately you want to report the meaningful parameter values, but you can't do that until you explain the model, and you can't do that until you explain the data being modeled. Therefore, recapitulate the data structure, reminding your reader of the predicted and predictor variables. Then describe the model, emphasizing the meaning of the parameters. This task of describing the model can be arduous for complex hierarchical models, but it is necessary and crucial if your analysis is to mean anything to your audience.
- Clearly describe and justify the prior: It is important to convince your audience that your prior is appropriate and does not predetermine the outcome. The prior should be amenable to a skeptical audience. The prior should be at least mildly informed to match the scale of the data being modeled. If there is copious previous research using very similar methods, it should not be ignored. Optionally, as mentioned again below, it may be helpful to try different priors and report the robustness of the posterior. When the goal is continuous parameter estimation, the posterior distribution is usually robust against reasonable changes in vague priors, but when the goal is model comparison (such as with inclusion coefficients) then the posterior probabilities of the models can be strongly affected by the choice priors.
- Report the MCMC details, especially evidence that the chains were converged and of sufficient length. Section 7.5 (p. 178) explained in detail how to examine the chains for convergence, and every program in this book produces diagnostic graphics of chains for at least some of the parameters. The programs all produce summaries of the parameters that include the ESS. Your report should indicate that the chains were checked for convergence, and your report should indicate the ESS of the relevant parameters. Usually this section of the report can be brief.
- Interpret the posterior: Many models have dozens or even hundreds of parameters, and therefore it is impossible to summarize all of them. The choice of which parameters or contrasts to report is driven by domain-specific theory and by the results themselves. You want to report the parameters and contrasts that are theoretically meaningful. You can report the posterior central tendency of a parameter and its HDI in text alone; histograms of posteriors are useful for the analyst to understand the posterior and for explanation in a textbook, but may be unnecessary in a concise report. Describe effects of shrinkage if appropriate. If your model includes interactions of predictors, be careful how you interpret lower-order effects. Finally, if you are using a ROPE for decisions, justify its limits.

25.1.2. Optional points

The following points are not necessarily crucial to address in every report, but should be considered. Whether or not to include these points depends on the particulars of the application domain, the points you want to make, and the audience to which the report is being addressed.

- Robustness of the posterior for different priors: When there is contention about the prior, it can be most convincing simply to conduct the analysis with different priors and demonstrate that the essential conclusions from the posterior do not change. This may be especially important when doing model comparisons, such as when using inclusion coefficients (or using Bayes factors to assess null values). Which priors should be used? Those that are meaningful and amenable to your audience, such as reviewers and editors of the journal to which the report is submitted.
- Posterior predictive check: By generating simulated data from the credible parameter values of the model, and examining the qualities of the simulated data, the veracity of the model can be further bolstered, if the simulated data do resemble the actual data. On the other hand, if the simulated data are discrepant from the actual data in systematic and interpretable ways, then the posterior predictive check can inspire new research and new models. For a perspective on posterior predictive checks, see the article by Kruschke (2013b) and Section 17.5.1 (among others) of this book.
- Power analysis: If there is only a weak effect in your results, what sample size would be needed to achieve some desired precision in the estimate? If you found a credibly nonzero difference, what was the retrospective power of your experiment and what is its replication power? This sort of information can be useful not only for the researcher's own planning, but it can also be useful to the audience of the report to anticipate potential follow-up research and to assess the robustness of the currently reported results. Chapter 13 described power analysis in depth.

25.1.3. Helpful points

Finally, to help science be cumulative, make your results available on the web:

- Post the raw data: There are at least two benefits of posting the original data. One benefit is that subsequent researchers can analyze the data with different models. New insights can be gained by alternative modeling interpretations. The longevity of the original research is enhanced. A second benefit is that if an exact or near-exact replication is conducted, the original data set can be concatenated with the new data set, to enhance sensitivity of the new data. Either way, your work gets cited!
- Post the MCMC sample of the posterior: There are at least two benefits of making the posterior publicly available. One is that other researchers can explore the posterior for effects and comparisons that were not in the report. Complex models have many parameters, and no single report can cover every possible perspective on the posterior

distribution. The longevity and impact of the research is thereby enhanced. A second benefit is that if subsequent researchers do follow-up research with a similar design and model, then the posted posterior can inform the prior of the subsequent analysis. Because the full posterior automatically incorporates all the covariations between all the parameters, the full posterior can be more useful than summaries of marginal distributions in a report. Either way, your work gets cited!

25.2. FUNCTIONS FOR COMPUTING HIGHEST DENSITY INTERVALS

HDIs have been used routinely throughout the book to describe the widths of distributions. Recall Figure 4.5 (p. 88) for examples, and review Figure 12.2 (p. 342) for an explanation of how HDIs differ from equal-tailed intervals. The present section provides details regarding how the HDIs are computed. The algorithm for computing an HDI on a grid approximation applies to any dimensionality and any shape distribution. The algorithms for computing an HDI of an MCMC sample or for a mathematical function apply only to single parameters with unimodal distributions. The R functions are defined in the file `DBDA2E-utilities.R`.

25.2.1. R code for computing HDI of a grid approximation

We can imagine the grid approximation of a distribution as a landscape of poles sticking up from each point on the parameter grid, with the height of each pole indicating the probability mass at that discrete point. We can imagine the highest density region by visualizing a rising tide: We gradually flood the landscape, monitoring the total mass of the poles that protrude above water, stopping the flood when 95% (say) of the mass remains protruding. The waterline at that moment defines the highest density region (e.g., Hyndman, 1996).

The function, listed below, finds the approximate highest density region in a somewhat analogous way. It uses one extra trick at the beginning, however. It first sorts all the poles in order of height, from tallest to shortest. The idea is to move down the sorted queue of poles until the cumulative probability has just barely exceeded 95% (or whatever mass is desired). The resulting height is the "waterline" that defines all points inside the highest density. See the comments in the top of the code for details of how to use the function.

```
HDIofGrid = function( probMassVec , credMass=0.95 ) {
  # Arguments:
  #   probMassVec is a vector of probability masses at each grid point.
  #   credMass is the desired mass of the HDI region.
  # Return value:
  #   A list with components:
  #   indices is a vector of indices that are in the HDI
  #   mass is the total mass of the included indices
```

```
  #   height is the smallest component probability mass in the HDI
  # Example of use: For determining HDI of a beta(30,12) distribution
  #   approximated on a grid:
  #   > probDensityVec = dbeta( seq(0,1,length=201) , 30 , 12 )
  #   > probMassVec = probDensityVec / sum( probDensityVec )
  #   > HDIinfo = HDIofGrid( probMassVec )
  #   > show( HDIinfo )
  sortedProbMass = sort( probMassVec , decreasing=TRUE )
  HDIheightIdx = min( which( cumsum( sortedProbMass ) >= credMass ) )
  HDIheight = sortedProbMass[ HDIheightIdx ]
  HDImass = sum( probMassVec[ probMassVec >= HDIheight ] )
  return( list( indices = which( probMassVec >= HDIheight ) ,
                mass = HDImass , height = HDIheight ) )
}
```

25.2.2. HDI of unimodal distribution is shortest interval

The algorithms for computing the HDI of an MCMC sample or of a mathematical function rely on a crucial property: For a unimodal probability distribution on a single variable, the HDI of mass M is the narrowest possible interval of that mass. Figure 25.1 illustrates why this is true. Consider the 90% HDI as shown. We construct another interval of 90% mass by moving the limits of the HDI to right, such that each limit is

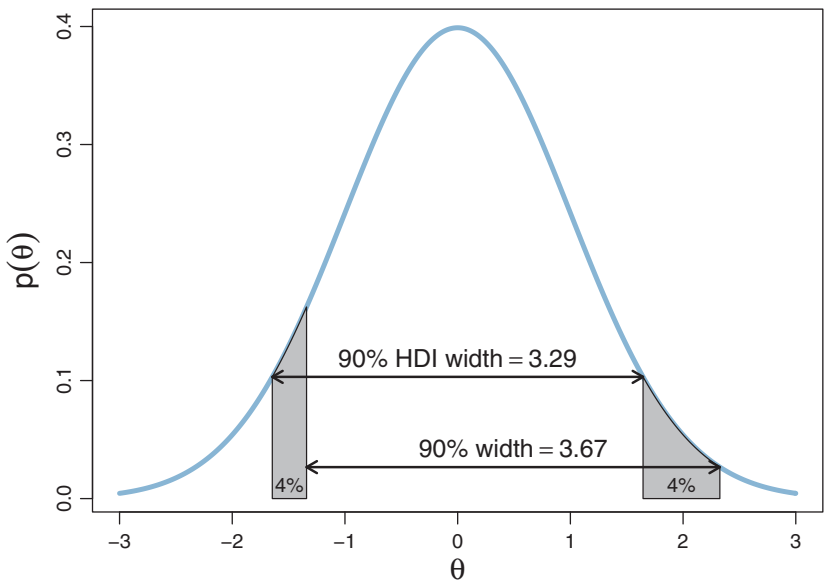

Figure 25.1 For a unimodal distribution, the HDI is the narrowest interval of that mass. This figure shows the 90% HDI and another interval that has 90% mass.

moved to a point that covers 4%, as marked in grey in Figure 25.1. The new interval must also cover 90%, because the 4% lost on the left is replaced by the 4% gained on the right.

Consider the grey regions in Figure 25.1. Their left edges have the same height, because the left edges are defined by the HDI. Their areas are the same, because, by definition, the areas are both 4%. Notice, however, that the left grey area is narrower than the right grey area, because the left area falls at a point where the distribution is increasing, but the right area falls at a point where the distribution is decreasing. Consequently, the distance between right edges of the two grey zones must be greater than the HDI width. (The exact widths are marked in Figure 25.1.) This argument applies for any size of grey zone, going right or left of the HDI, and for any mass HDI. The argument relies on unimodality, however.

Given the argument and diagram in Figure 25.1, it is not too hard to believe the converse: For a unimodal distribution on one variable, for any mass M, the interval containing mass M that has the narrowest width is the HDI for that mass. The algorithms described below are based on this property of HDIs. The algorithms find the HDI by searching among candidate intervals of mass M. The shortest one found is declared to be the HDI. It is an approximation, of course. See Chen and Shao (1999) for more details, and Chen, He, Shao, and Xu (2003) for dealing with the unusual situation of multimodal distributions.

25.2.3. R code for computing HDI of a MCMC sample

It is important to remember that an MCMC sample is only a random and noisy representation of the underlying distribution, and therefore the HDI found from the MCMC sample is also noisy. For details, review the discussion of HDI approximation error that accompanies Figures 7.13 (p. 185) and 7.14 (p. 186). Below is the code for finding the HDI of an MCMC sample. It is brief and hopefully self-explanatory after the discussion of the previous section.

```
HDIofMCMC = function( sampleVec , credMass=0.95 ) {
  # Computes highest density interval from a sample of representative values,
  #   estimated as shortest credible interval.
  # Arguments:
  #   sampleVec
  #     is a vector of representative values from a probability distribution.
  #   credMass
  #     is a scalar between 0 and 1, indicating the mass within the credible
  #     interval that is to be estimated.
  # Value:
  #   HDIlim is a vector containing the limits of the HDI
  sortedPts = sort( sampleVec )
  ciIdxInc = ceiling( credMass * length( sortedPts ) )
```

```
  nCIs = length( sortedPts ) - ciIdxInc
  ciWidth = rep( 0 , nCIs )
  for ( i in 1:nCIs ) {
    ciWidth[ i ] = sortedPts[ i + ciIdxInc ] - sortedPts[ i ]
  }
  HDImin = sortedPts[ which.min( ciWidth ) ]
  HDImax = sortedPts[ which.min( ciWidth ) + ciIdxInc ]
  HDIlim = c( HDImin , HDImax )
  return( HDIlim )
}
```

25.2.4. R code for computing HDI of a function

The function described in this section finds the HDI of a unimodal probability density function that is specified mathematically in R. For example, the function can find HDI's of normal densities or of beta densities or of gamma densities, because those densities are specified as functions in R.

What the program accomplishes is just a search of HDI's, converging to the shortest one, but it does this by using some commands and R functions that may not have been used elsewhere in the book. One function that the program uses is the inverse cumulative density function (ICDF) for whatever probability distribution is being targeted. We have seen one case of an ICDF previously, namely the probit function, which is the inverse of the cumulative-density function for the normal distribution. In R, the ICDF of the normal is the `qnorm(x)` function, where the argument `x` is a value between zero and one. The program for finding the HDI takes, as one of its arguments, R's name for the ICDF of the function. For example, if we want to find an HDI of a normal density, we pass in `ICDFname="qnorm"`.

The crucial function called by the program is R's `optimize` routine. The `optimize` routine searches and finds the minimum of a specified function over a specified domain. In the program below, we define a function called `intervalWidth` that returns the width of the interval that starts at `lowTailPr` and has 95% mass. This `intervalWidth` function is repeatedly called from the `optimize` routine until it converges to a minimum.

```
HDIofICDF = function( ICDFname , credMass=0.95 , tol=1e-8 , ... ) {
  # Arguments:
  #   ICDFname is R's name for the inverse cumulative density function
  #     of the distribution.
  #   credMass is the desired mass of the HDI region.
  #   tol is passed to R's optimize function.
  # Return value:
  #   Highest density interval (HDI) limits in a vector.
  # Example of use: For determining HDI of a beta(30,12) distribution, type
  #   > HDIofICDF( qbeta , shape1 = 30 , shape2 = 12 )
```

```
  #   Notice that the parameters of the ICDFname must be explicitly named;
  #   e.g., HDIofICDF( qbeta , 30 , 12 ) does not work.
  # Adapted and corrected from Greg Snow's TeachingDemos package.
  incredMass =  1.0 - credMass
  intervalWidth = function( lowTailPr , ICDFname , credMass , ... ) {
    ICDFname( credMass + lowTailPr , ... ) - ICDFname( lowTailPr , ... )
  }
  optInfo = optimize( intervalWidth , c( 0 , incredMass ) , ICDFname=ICDFname ,
                      credMass=credMass , tol=tol , ... )
  HDIlowTailPr = optInfo$minimum
  return( c( ICDFname( HDIlowTailPr , ... ) ,
             ICDFname( credMass + HDIlowTailPr , ... ) ) )
}
```

25.3. REPARAMETERIZATION

There are situations in which one parameterization is intuitive to express a distribution, but a different parameterization is required for mathematical convenience. For example, we may think intuitively of the standard deviation of a normal distribution, but have to parameterize the distribution in terms of the precision (i.e., reciprocal of the variance). The question is, if we express a probability distribution on one scale, what is the equivalent distribution on a transformed scale? The answer is not difficult to figure out, especially for single parameters.

Let the "destination" parameter be denoted θ, and suppose that $\theta = f(\phi)$ for the "source" parameter ϕ, with a monotonic and differentiable function f. Let the probability distribution on ϕ be denoted $p(\phi)$. Then the corresponding probability distribution on θ is

$$p(\theta) = \frac{p(f^{-1}(\theta))}{|f'(f^{-1}(\theta))|} \tag{25.1}$$

where $f'(\phi)$ is the derivative of f with respect to ϕ.

Here's why. Consider a small (actually infinitesimal) interval under the distribution $p(\phi)$, at a particular value ϕ. Call the width of the interval $d\phi$. The probability mass in that interval is the product of the density and the width: $p(\phi) \cdot d\phi$. We want to construct a probability density on θ, which we denote $p(\theta) = p\big(f(\phi)\big)$, that has the same probability mass in the corresponding interval at $\theta = f(\phi)$. The width of the corresponding interval on θ is $d\theta = d\phi \cdot |f'(\phi)|$ because, by definition of the derivative, $f'(\phi) = d\theta/d\phi$. So, the probability mass in that interval is $p(\theta) \cdot d\theta = p\big(f(\phi)\big) \cdot d\phi \cdot |f'(\phi)|$. Therefore, to equate the probability masses in the corresponding intervals, we require that $p\big(f(\phi)\big) \cdot d\phi \cdot |f'(\phi)| = p(\phi) \cdot d\phi$, which, when rearranged, yields $p\big(f(\phi)\big) = p(\phi) \,/\, |f'(\phi)|$, which is Equation 25.1.

25.3.1. Examples

An example was given in Figure 21.11 (p. 640), which had a normal distribution on parameter β_0 transformed to a distribution over parameter μ via a logistic function. The question was, What is the implied distribution on μ? Footnote 2 on p. 639 explained the application of Equation 25.1.

As another example, we can apply Equation 25.1 to the case in which $\theta = f(\phi) = \text{logistic}(\phi) = 1/[1 + \exp(-\phi)]$, and the distribution on ϕ is given by $p(\phi|a, b) = \big(f(\phi)\big)^a \big(1 - f(\phi)\big)^b / B(a, b)$. An example of this distribution, for $a = 1$ and $b = 1$, is shown in the top-left panel of Figure 25.2. Notice that the derivative of the logistic function f is $f'(\phi) = \exp(-\phi)/[1 + \exp(-\phi)]^2 = f(\phi)\,(1 - f(\phi)) = \theta(1 - \theta)$. Therefore, according to Equation 25.1, the equivalent probability density at $\theta = f(\phi)$ is $p(\theta) = p(\phi)/f'(\phi) = \theta^a\,(1 - \theta)^b / [\theta(1 - \theta)B(\theta|a, b)] = \theta^{(a-1)}\,(1 - \theta)^{(b-1)} / B(a, b) = \text{beta}(\theta|a, b)$. The upper row of Figure 25.2 shows this situation when $a = b = 1$. An intuitive way to think of this situation is that the probability on ϕ is dense near $\phi = 0$, but that is exactly where the logistic transformation stretches the distribution. On the other hand, the probability on ϕ is sparse at large positive or large negative values, but that is exactly where the logistic transformation compresses the distribution.

As another example, consider a case in which $\theta = f(\psi) = 1 - \exp(-\psi)$, with the probability density $p(\psi) = \exp(-\psi)$. Notice that the derivative of the transformation is $f'(\psi) = \exp(-\psi)$, and therefore the equivalent density at $\theta = f(\phi)$ is $p\big(f(\psi)\big) = p(\psi)/f'(\psi) = 1$. In other words, the equivalent density on θ is the uniform density, as shown in the lower row of Figure 25.2.

As a final example, Figure 25.3 shows a uniform distribution on standard deviation transformed to the corresponding distribution on precision. By definition, precision is the reciprocal of squared standard deviation.

25.3.2. Reparameterization of two parameters

When there is more than one parameter being transformed, the calculus becomes a little more involved. Suppose we have a probability density on a two-parameter space, $p(\alpha_1, \alpha_2)$. Let $\beta_1 = f_1(\alpha_1, \alpha_2)$ and $\beta_2 = f_2(\alpha_1, \alpha_2)$. Our goal is to find the probability density $p(\beta_1, \beta_2)$ that corresponds to $p(\alpha_1, \alpha_2)$. We do this by considering infinitesimal corresponding regions. Consider a point α_1, α_2. The probability mass of a small region near that point is the density at that point times the area of the small region: $p(\alpha_1, \alpha_2)\, d\alpha_1\, d\alpha_2$. The corresponding region in the transformed parameters should have the same mass. That mass is the density at the transformed point times the area of the region mapped to from the originating region. In vector calculus textbooks, you can find discussions demonstrating that the area of the mapped-to region

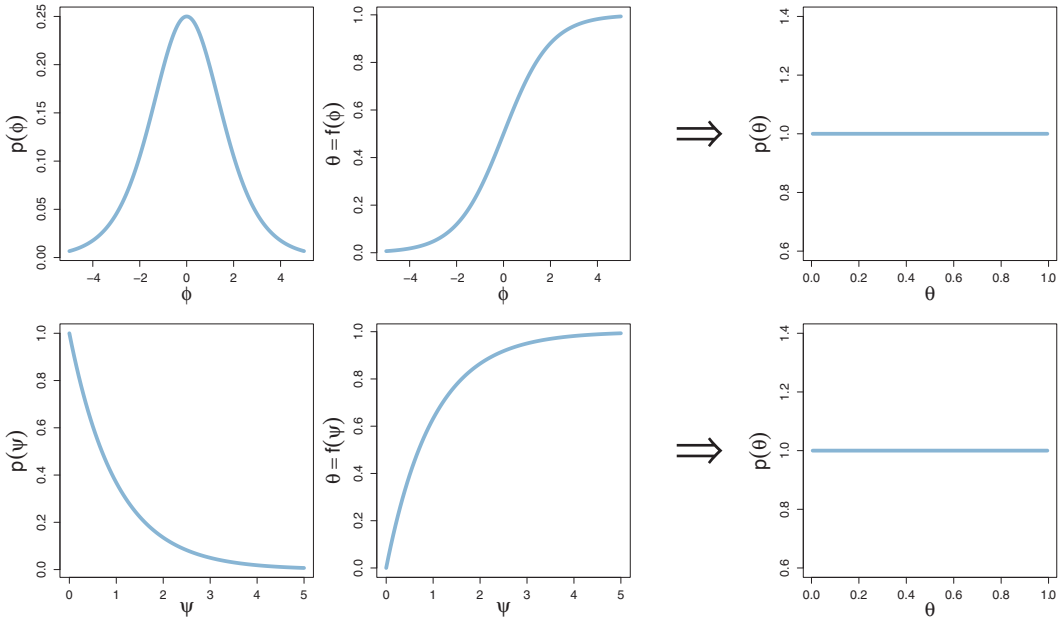

Figure 25.2 *Top row:* A reparameterization that maps a peaked distribution over $\phi \in [-\infty, +\infty]$ to a uniform distribution over $\theta \in [0, 1]$. *Bottom row:* A reparameterization that maps a descending exponential distribution over $\psi \in [0, +\infty]$ to a uniform distribution over $\theta \in [0, 1]$.

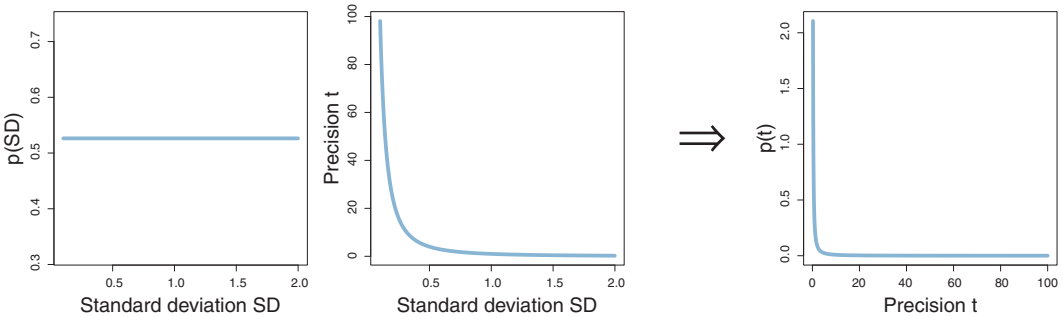

Figure 25.3 A uniform distribution on standard deviation, mapped to the corresponding distribution on precision.

is $|\det(J)|\, d\alpha_1\, d\alpha_2$ where J is the Jacobian matrix: $J_{r,c} = df_r(\alpha_1, \alpha_2)/d\alpha_c$ and $\det(J)$ is the determinant of the Jacobian matrix. Setting the two masses equal and rearranging yields $p(\beta_1, \beta_2) = p(\alpha_1, \alpha_2)/|\det(J)|$. You can see that Equation 25.1 is a special case. As we have had no occasions to apply this transformation, no examples will be provided here.

But the method has been mentioned for those intrepid souls who may wish to venture into the wilderness of multivariate probability distributions with nothing more than pen and paper.

25.4. CENSORED DATA IN JAGS

In many situations some data are censored, which means that their values are known only within a certain range. For example, in response-time experiments, the experimenter is usually willing to wait for a response for only a limited duration such as 5 s, beyond which the measurement trial is terminated. If the responder was "on task" but did not respond before the 5 s limit, the datum is still informative, because it means the task took longer than 5 s. Censored data should not be discarded because the remaining data are biased to be below the censoring cut-off in this case. An analogous situation occurs when measuring life span after a treatment: Often the researcher cannot wait for all subjects to expire, and the subjects still surviving after the limited measurement duration contribute censored data. Data can also be censored on the low side, for example when a measuring device has a lower limit. Sometimes data can be interval censored, which means that there are some intervals in which we do not know the exact value, only the bounds on the interval. A contrived example of interval censoring is when we measure the distance that a putted miniature-golf ball rolls across the carpet, with its path partially covered by a tunnel. If the ball stops within the tunnel, all we know is that its distance is somewhere between the entrance and exit of the tunnel.

In all the examples of this section, we are assuming that the censoring is independent of the variable being measured. For example, we assume that the experimenter's 5 s time limit does not affect the responder's response time, and the responder's time does not affect the experimenter's limit. Similarly, we assume that tunnel does not affect the distance travelled by the ball, and we assume that the distance traveled by the ball does not affect the position of the tunnel.

To illustrate why it is important to include censored data in the analysis, consider a case in which $N = 500$ values are generated randomly from a normal distribution with $\mu = 100$ and $\sigma = 15$. Suppose that values above 106 are censored, as are values in the interval between 94 and 100. For the censored values, all we know is the interval in which they occurred, but not their exact value. Figure 25.4 shows results from two analyses of the data. The upper quartet of panels in Figure 25.4 shows results when the censored data are excluded from the analysis. There are only $N = 255$ uncensored values, and they have a mean far less than 100. The posterior estimate of μ and σ are both less than the underlying true values. The lower quartet of Figure 25.4 shows results when the censored data are retained in the analysis. The posterior estimate of μ and σ is very accurate in this case, despite the uncertainty in the censored data values.

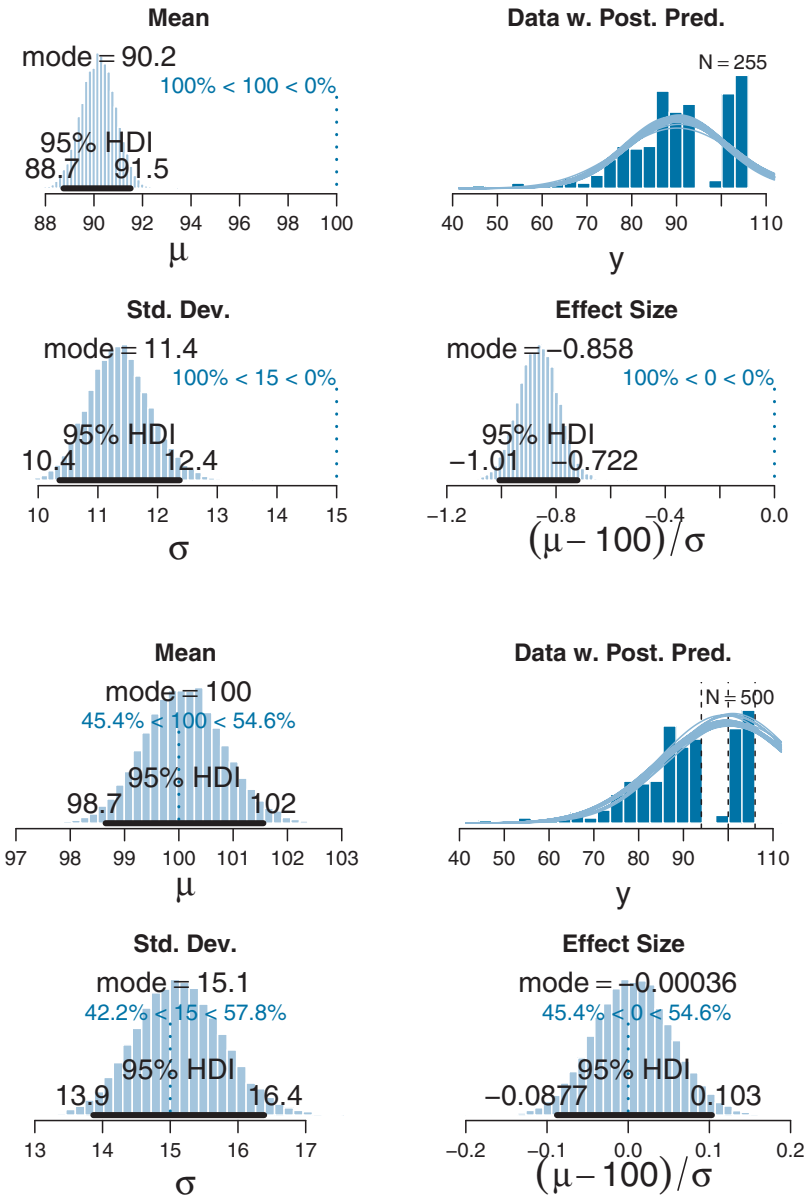

Figure 25.4 Data randomly generated from a normal distribution with $\mu = 100$ and $\sigma = 15$, then censored between 94 and 100 and above 106. *Upper quartet:* Censored data omitted from analysis; parameter estimates are too small. *Lower quartet:* Censored data imputed in known bins; parameter estimates are accurate.

The remainder of this section explains how the analysis of censored data is implemented in JAGS. The first crucial feature is the coding of the data. The censored values need to indicate not merely that they were censored, but also which censoring interval they came from, and the thresholds of the intervals. To save space, the intervals are here called "bins." Some lines from the data file for the example in Figure 25.4 follow:

```
   y    ybin  thresh1  thresh2  thresh3   #   yOrig
 79.36   0      94.0    100.0    106.0    #   79.36
 82.97   0      94.0    100.0    106.0    #   82.97
   NA    1      94.0    100.0    106.0    #   95.61
   NA    1      94.0    100.0    106.0    #   99.87
103.28   2      94.0    100.0    106.0    #  103.28
104.40   2      94.0    100.0    106.0    #  104.40
   NA    3      94.0    100.0    106.0    #  125.74
   NA    3      94.0    100.0    106.0    #  130.60
```

The column named `y` contains the main data. Notice that censored values are coded as `NA`. Each row of the data file also specifies the bin in which the `y` value falls, in a column titled `ybin`. Notice that the bin numbering for JAGS begins at 0, not at 1. The thresholds that define the bin limits are specified in the next columns. In this example there are three thresholds, at 94.0, 100.0, and 106.0. Values of `y` less than the first threshold have `ybin`=0. Values of `y` greater than the first threshold but less than the second threshold have `ybin`=1. And so on. In principle, the thresholds could be different for each datum (as long as they are independent of the data), and therefore the threshold values are repeated on every row of the data file. For conceptual clarity, the original random values from the normal distribution are included in the column labeled `yOrig`, but no real data file would have such a column, of course.

There are two aspects to understanding how JAGS models censored data. First, JAGS simultaneously describes both `y` and `ybin`. In this sense JAGS uses a multivariate model, predicting two variables from its underlying parameters. The key lines in the model specification look like this:

```
y[i] ~ dnorm( mu , 1/sigma^2 )
ybin[i] ~ dinterval( y[i] , threshMat[i, ] )
```

The first line above says that the `y[i]` value comes from a normal distribution with mean `mu` and standard deviation `sigma`. The second line above says that the `ybin[i]` value comes from an "interval" distribution, for which the first argument is the value `y[i]` and the second argument is a vector of the threshold values, here denoted `threshMat[i,]`. If you think of the interval distribution as a generator of `ybin` values, then it produces `ybin`=0 when `y` is less than the lowest threshold in `threshMat[i,]`, it produces `ybin`=1 when `y` is between the lowest threshold and next lowest threshold in `threshMat[i,]`, and so on. If you think of the interval distribution as a probability distribution, then it has

probability 1.0 whenever the value of ybin is appropriate for the value of y relative to the values of the thresholds in threshMat[i,], and the interval distribution has probability 0.0 whenever the value of ybin is not appropriate for the value of y relative to the values of the thresholds in threshMat[i,]. If every y[i] value were present (i.e., uncensored) then the line that specifies the interval distribution would have no effect on the model because its probability would always be 1.0.

The second aspect to understanding how JAGS models censored data is the fact that when JAGS encounters a missing data value, JAGS automatically imputes a random value generated from the model and the credible parameter values at that step in the MCMC chain. In other words, JAGS treats missing data values as if they were parameters, and the whole model (informed by the other data) acts as a prior distribution for generating the missing parameter. If JAGS knew only that a missing data value y[i] came from a normal distribution, then JAGS would impute a random value sampled from the normal distribution. But JAGS knows also that the missing data value y[i] must satisfy the interval distribution, and therefore the imputed y[i] value must also come from the appropriate bin.

The full model specification is otherwise familiar:

```
model {
  for ( i in 1:Ntotal ) {
    y[i] ~ dnorm( mu , 1/sigma^2 )
    ybin[i] ~ dinterval( y[i] , threshMat[i, ] )
  }
  mu ~ dnorm( meanY , 1/(100*sdY)^2 )
  sigma ~ dunif( sdY/1000 , sdY*1000 )
}
```

The only other essential for implementing the model in JAGS is initializing the chains for the missing values of y. While JAGS can initialize many parameters automatically, the imputed data in this model structure need to be initialized explicitly. The data vector, y, has both known and imputed values, and we only need to initialize the imputed components, leaving the known values at the values supplied in the data file. There is a complementarity between the data vector y, which has NA at imputed components, and the initial value for the chain y, which has NA at known components. A similar approach was taken to initializing thresholds in the thresholded cumulative-normal model of Section 23.2.1 (p. 676). The following R commands set up initial values for the y chain, denoted yInit:

```
yInit = rep( NA , length(y) )                    # Define placeholder.
for ( i in 1:length(y) ) {                       # For each datum,
  if ( is.na(y[i]) ) {                           # if y is censored, then:
```

```
    if ( ybin[i]==0 ) {                          # if it's from the lowest bin
      yInit[i] = threshMat[i,1]-1                # initialize at below low thresh;
    } else if ( ybin[i]==ncol(threshMat) ) {     # if it's from the highest bin
      yInit[i] = threshMat[i,ncol(threshMat)]+1  # initialize at above high thresh;
    } else {                                     # else initialize at middle of bin.
      yInit[i] = (threshMat[i,ybin[i]]+threshMat[i,ybin[i]+1])/2
    }
  }
}
initsList = list( y=yInit )  # Assemble into the list later sent to JAGS.
```

The model and its supporting functions can be found in the file named `Jags-YmetBinned-Xnom1grp-MnormalInterval.R`, and the high-level script that calls the functions is `Jags-YmetBinned-Xnom1grp-Mnormal Interval-Example.R`.

25.5. WHAT NEXT?

If you have made it this far and you are looking for more, you might peruse posts at my blog, http://doingbayesiandataanalysis.blogspot.com/, and search there for topics that interest you. When you are ready for more applications of frequently-used statistical models, consider the book by Ntzoufras (2009), which provides many examples in R and WinBUGS. The WinBUGS examples can be easily converted to JAGS. The book by Gelman et al. (2013) uses Stan and covers many advanced topics including nonparametric models such as Gaussian process and Dirichlet process models. Finally, the book by Lee and Wagenmakers (2014) examines applications of WinBUGS to models of cognitive processes such as memory, categorization, and decision making.