9

Issues in Modelling

A strength of the Bayesian graphical modelling techniques of BUGS is the way they can represent the typical complexities of real data. This chapter explains various generic issues encountered in data analysis and how they can be addressed in BUGS. For example, data commonly include missing values and measurement errors. A realistic model may need to account for censoring, truncation, grouping, rounding, or constraints on parameters, or use a sampling or prior distribution not already included in BUGS. We also discuss prediction, controlling "feedback" in graphical models, classical bootstrap estimation, and expressing uncertainty surrounding "ranks" or positions in a league table. Each of the techniques we describe may be deployed as part of *any* model in BUGS, with typically only a few extra lines of code.

9.1 Missing data

Missing data are common and there is an extensive literature covering a wide variety of methods for dealing with the problem. Comprehensive textbooks on the topic include Little and Rubin (2002), Molenberghs and Kenward (2007), and Daniels and Hogan (2008). Missing values in BUGS are denoted by `NA` in the data set, and from a Bayesian perspective, these are treated as additional unknown quantities for which a posterior distribution can be estimated. Hence the Bayesian approach makes no fundamental distinction between missing data and unknown model parameters. We just need to specify an appropriate joint model for the observed and missing data and model parameters, and BUGS will generate posterior samples of the model parameters and missing values in the usual way using MCMC.

The appropriateness of a particular missing data model is dependent on the mechanism that leads to the missing data and the pattern of the missing data. It also makes a difference whether we are dealing with missing responses or missing covariates (or both). Following Rubin (1976), missing data are generally classified into three types: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). Informally, MCAR occurs when the probability of missingness does not depend on observed or unobserved data, in the less restrictive MAR it depends only on the observed data, and when neither MCAR nor MAR holds, the data are MNAR. Under an MCAR or MAR assumption, it is not usually necessary to specify a model for the missing data *mechanism* in order to make valid inference about parameters of the observed data likelihood, in which case the missing data mechanism is termed *ignorable*. In the case of MNAR missingness, the fact that a given value is missing tells us something about what that value might have been. In this case the missing data mechanism is *informative* and we must specify a model for it. There are two main approaches to this, using either a pattern mixture model or a selection model (Daniels and Hogan, 2008). In either case, the parameters of such a model cannot be uniquely inferred from the data, and so informative priors or parameter constraints are typically required. Inferences can thus be sensitive to the choices made — see Best et al. (1996) and Mason et al. (2012) for detailed discussions in the case of selection models. In the following examples we illustrate how to implement some specific models for missing response or missing covariate data in BUGS that make different assumptions about the missing data mechanism. A comprehensive discussion of a wide range of Bayesian missing data models can be found in Daniels and Hogan (2008).

9.1.1 Missing response data

For *ignorable* missing response data, we can chose to remove it from the data set, but often this is inconvenient. If we simply denote the value as missing (`NA`) in the dataset, then BUGS will automatically generate values from its posterior predictive distribution — see §9.2 — and inferences on the parameters will be as if we had deleted that response.

Example 9.1.1. *Growth curve (continued): ignorable missing response data mechanism*
We look again at the growth data from a single rat previously considered as an example of regression analysis (Example 6.1.1). We assume that the final datapoint (actually 376 g) is missing. If we assume that the chance of a value being missing does not directly depend on the true underlying weight, then an identical regression model can be adopted and only the data file changed.

```
for (i in 1:5) {
  y[i]       ~ dnorm(mu[i], tau)
  mu[i]     <- alpha + beta*(x[i] - mean(x[]))
}
alpha        ~ dflat()
.....
list(y = c(177,236,285,350,NA), x = c(8,15,22,29,36))

node   mean  sd     MC error 2.5%  median 97.5% start sample
alpha  290.3 7.029  0.06709  279.4 290.4  300.8 4001  10000
```

```
beta   8.104 0.8881 0.007908 6.79  8.11   9.355 4001  10000
sigma2 188.1 2915.0 58.29    5.72  29.08  926.0 4001  10000
mu[5]  403.8 16.74  0.158    377.6 404.0  427.7 4001  10000
y[5]   403.6 20.52  0.2122   371.5 404.0  434.2 4001  10000
```

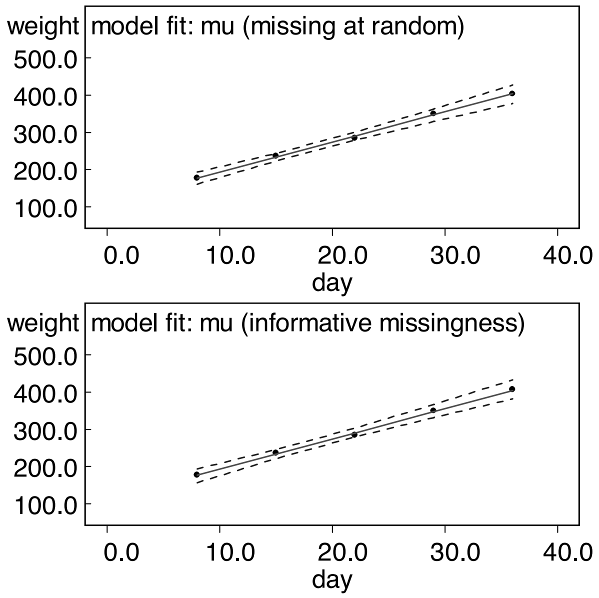

FIGURE 9.1
Model fits for rat 9's data with final observation missing. The plotted point (•) corresponding to the missing value `y[5]`, at 36 days, is the posterior mean. The interval plotted at `x[5]` is for `mu[5]`, not `y[5]`. Top: `y[5]` missing at random. Bottom: informative missingness for `y[5]`.

The model fit is shown at the top of Figure 9.1. The estimated value for the missing data point `y[5]` lies on the fitted line; the 95% credible interval is wider than that for `mu[5]` (see table above), since it allows for the uncertainty about `mu[5]` as well as for the sampling error σ $(= \tau^{-1/2})$, and uncertainty about σ.

Example 9.1.2. *Growth curve (continued): informative missing response data mechanism*
Again we assume the final observation is missing, but now we also assume that the chance of an observation being missing depends on the true weight: specifically

the odds of being missing increase by 2% for every extra gram of weight. The data now has to include an indicator `miss[]` for whether an observation is missing or not, and our assumption about the missing data mechanism is specified using a logistic selection model for the probability of missingness, with b=log(1.02).

```
  for (i in 1:5) {
    y[i]          ~ dnorm(mu[i], tau)
    mu[i]        <- alpha + beta*(x[i] - mean(x[]))

    # selection model for missing data mechanism
    miss[i]       ~ dbern(p[i])
    logit(p[i]) <- a + b*(y[i]-250)
  }
  a               ~ dlogis(0, 1)
  b              <- log(1.02)
  .....
  list(y = c(177,236,285,350,NA), x = c(8,15,22,29,36),
        miss = c(0,0,0,0,1))
```

Here we specify a logistic prior for a, equivalent to a uniform prior on the probability of an observation with a true value of 250 g being missing (see §5.2.5).

```
node   mean   sd     MC error 2.5%   median 97.5%   start sample
a      -1.973 1.082  0.01081  -4.262 -1.913 0.01912 4001  10000
alpha  291.4  13.13  0.3402   280.5  290.6  303.5   4001  10000
beta   8.228  1.314  0.03709  7.035  8.133  9.677   4001  10000
sigma2 433.0  4537.0 177.5    5.755  30.73  1063.0  4001  10000
mu[5]  406.6  27.89  0.8487   382.7  404.6  435.0   4001  10000
y[5]   408.4  38.23  1.368    378.6  404.9  447.1   4001  10000
```

The assumption about the missing data mechanism has raised the estimated missing weight from 404 to 408 g, and the posterior mean gradient from 8.10 to 8.23. The fact that the final data point is missing suggests that it has a larger value than it might have had if it were missing at random. Note that the missing weight is no longer estimated on the fitted line but slightly above it — see Figure 9.1 (bottom). In practice, we would want to examine sensitivity to different assumptions about the missing data mechanism, particularly to the value of b. It is also possible to treat b as random and specify a prior distribution for it, although posterior learning about the parameters of the selection model is heavily dependent on model assumptions (see Mason et al. 2012). Note that with more complex examples, reasonably tight priors and careful choice of initial values for the parameters of the selection model may be needed to avoid convergence problems and possible crashes of the MCMC sampling algorithms.

9.1.2 Missing covariate data

In the case of missing covariates, again `NA` can be specified for each missing value. However, the difference between missing responses and missing covariates is that we would not have specified a prior distribution or model for the covariates if they had been fully observed. Hence we must introduce a model or prior for the missing values, even if the missing data mechanism can be assumed to be ignorable. One option is to specify a prior distribution with common unknown parameters for both observed and unobserved values of the relevant covariate and then priors for the parameters of this distribution. The observed values of the covariate will contribute to the estimation of the unknown parameters, which, in turn, will inform about the missing values. This approach is only valid if the missing and observed covariate values can be assumed to be *exchangeable*, which is a mathematical formalisation of the assumption that a group of quantities is *similar* in some sense. Such assumptions are further discussed in Chapter 10. We illustrate this approach below.

Example 9.1.3. *Dugongs (continued): ignorable missing covariate mechanism*
We look again at version 3 of the non-linear growth curve model for the lengths of 27 dugongs, previously considered as an example of regression analysis (Example 6.3.1). In the current example we assume that the ages of four of the dugongs were not recorded. In specifying a prior distribution for the missing ages we should bear in mind that the growth curve is only meaningful for non-negative values of age. Here we constrain each unknown age to be positive by assuming that all ages arise from a log-normal distribution (`dlnorm()` — see Appendix C.2) with common unknown parameters. A more appropriate assumption might be a truncated normal distribution, truncated at zero. Note, however, that we would not be able to use the `I(,)` construct to truncate the distribution, since the parameters of that distribution would be unknown — see §9.6.2 and Appendix A.2.2 for further details, including alternative ways of specifying truncated distributions.

```
for(j in 1:27) {
  y[j]       ~ dnorm(mu[j], tau)
  mu[j]     <- alpha - beta*pow(gamma, x[j])
  # prior on covariate
  x[j]       ~ dlnorm(mu.x, p.x)
}
...
# priors on mean and precision of covariate model
mu.x         ~ dunif(-10, 10)
p.x         <- 1/pow(sd.x, 2)
sd.x         ~ dunif(0, 10)
...
list(x = c(1.0,1.5,1.5,NA,2.5,4.0,5.0,5.0,NA,8.0,8.5,9.0,
           9.5,9.5,10.0,12.0,12.0,13.0,NA,14.5,15.5,15.5,
          16.5,17.0,NA,29.0,31.5),
     y = c(1.80,1.85,1.87,1.77,2.02,2.27,2.15,2.26,2.47,
          2.19,2.26,2.40,2.39,2.41,2.50,2.32,2.32,2.43,
          2.47,2.56,2.65,2.47,2.64,2.56,2.70, 2.72,2.57))
```

```
node   mean   sd      MC error 2.5%    median  97.5%  start sample
alpha  2.649  0.06845 0.001528 2.527   2.645   2.797  4001  50000
beta   0.9604 0.07883 9.338E-4 0.8136  0.9584  1.116  4001  50000
gamma  0.8661 0.03251 7.006E-4 0.7926  0.8707  0.9152 4001  50000
mu.x   2.096  0.2087  9.875E-4 1.689   2.094   2.512  4001  50000
sd.x   1.048  0.1722  0.001139 0.7735  1.027   1.444  4001  50000
sigma  0.0955 0.01605 1.407E-4 0.07019 0.09341 0.1325 4001  50000
x[4]   1.395  0.7006  0.004034 0.3363  1.299   3.035  4001  50000
x[9]   16.7   18.08   0.1805   6.831   12.77   54.61  4001  50000
x[19]  17.52  22.87   0.2601   6.821   12.88   59.72  4001  50000
x[25]  38.5   39.62   0.3918   12.3    28.05   127.9  4001  50000
```

There is considerable uncertainty regarding the "true" values of the missing covariates, particularly for dugongs 9, 19, and 25. The actual values of the four missing ages are 1.5, 7, 13, and 22.5, and all of these are included within the estimated 95% credible intervals, although for dugong 9 this is only just the case. This reflects the influence of the response value, y, on the posterior distribution of each missing x — dugong 9 was long for its age (y[9] = 2.47) and hence values of age somewhat larger than the actual value are consistent with the fitted model. In fact, dugongs 9 and 19 had identical lengths, and so the posterior distributions for x[9] and x[19] are identical to within sampling error.

If there are other fully observed covariates in the regression model of interest, the previous approach will not account for correlation between these and the covariate being imputed. In this case, a better option is to specify a regression model to impute the missing covariates as a function of other covariates. This model may include covariates not in the main model of interest and is similar in spirit to the two-stage multiple imputation (MI) approach of Rubin (1987). As with standard MI, variables that are predictive of both the missing covariate itself *and* of the missing data mechanism should be included in the imputation model. When multiple covariates have missing values, it is also important to reflect the dependence structure of the covariates in the imputation model.

Example 9.1.4. *Birthweight: regression model for imputing missing covariates*
Trihalomethanes (THM) are a chemical byproduct of the treatment process used to disinfect the public water supply in the UK. Molitor et al. (2009) analyse the association between THM levels in domestic tap water and the risk of giving birth to a low birthweight (< 2.5 kg) baby. They use data from the UK National Births Register on maternal age, baby's gender and birthweight, and use the mother's

residential postcode to link modelled estimates of average THM levels in the water supply zone of residence to each birth. Maternal smoking and ethnicity are known risk factors for low birthweight and are potential confounders of the THM effect due to their spatial patterning, which correlates with spatial variations in THM levels. Smoking and ethnicity are not recorded in the birth register but are available for a subset of the mothers who participated in a national birth cohort study. Molitor et al. (2009) build a full Bayesian model to impute the missing smoking and ethnicity indicators for mothers in the birth register who did not participate in the cohort study and to simultaneously estimate the regression of low birthweight on THM levels adjusted for confounders.

Here we use simulated data that mimics a slightly simplified version of this problem. The model of interest is a logistic regression of the low birthweight indicator `lbw` on binary indicators of THM level > 60 µg/L (`THM`), male baby (`male`), non-white maternal ethnicity (`eth`), maternal smoking during pregnancy (`smk`), and deprived local area (`dep`). `smk` and `eth` are recorded for 20% of mothers but are missing for the remaining 80%; all other variables are fully observed. To impute the missing covariate values, we build a bivariate regression model for `smk` and `eth` assuming correlated errors. Since these are both binary indicators, we use multivariate probit regression (Chib and Greenberg, 1998) in which `smk` and `eth` are equal to thresholded values of a bivariate normal latent variable `Z`,

$$Z_i = (Z_{i1}, Z_{i2})' \sim \mathsf{MVN}(\mu_i, \Omega), \quad i = 1, \ldots, n;$$
$$smk_i = I(Z_{i1} \geq 0); \quad eth_i = I(Z_{i2} \geq 0),$$

where Ω must be in correlation form for identifiability reasons. The elements of $\mu_i = (\mu_{i1}, \mu_{i2})'$ are modelled as independent linear functions of covariates, which include the other variables in the regression model for `lbw` plus area-level measures of the proportion of the population who smoke (`area.smk`) and who are non-white (`area.eth`). Unlike standard multiple imputation, it is *not* necessary to include the response variable from the regression model of interest (`lbw` in this case) in the covariate imputation model, since information about `lbw` is automatically propagated via feedback from the assumed regression model of `lbw` on `smk` and `eth`. Likelihood information about the observed values of `smk` and `eth` is included in the imputation model by specifying bounds on Z_i such that $Z_{i1} \in (-\infty, 0)$ if $smk_i = 0$, $Z_{i1} \in [0, \infty)$ if $smk_i = 1$, $Z_{i2} \in (-\infty, 0)$ if $eth_i = 0$, and $Z_{i2} \in [0, \infty)$ if $eth_i = 1$. If smk_i and eth_i are missing, the corresponding bounds on Z_{i1} and Z_{i2} are set to $(-\infty, \infty)$. In BUGS this is done using the `I(lower,upper)` notation (§9.6) and including vectors giving values of the lower and upper bounds in the data file. Since a value of $\pm\infty$ cannot be specified in the data file, we instead use an arbitrarily large value relative to the scale of the latent Z variable (say $\pm$ 10). Initial values for the parameters of the regression model of interest and for the imputation model need to be chosen carefully to ensure that they both provide compatible information about the missing covariate values; strongly conflicting initial values can cause the MCMC samplers in BUGS to crash.

```
for (i in 1:n) {
  lbw[i]          ~ dbern(p[i])
  logit(p[i])    <- beta[1] + beta[2]*THM[i] +
                    beta[3]*male[i] + beta[4]*dep[i] +
                    beta[5]*smk[i] + beta[6]*eth[i]
}
for (k in 1:6) {
  beta[k]         ~ dnorm(0, 0.0001)
}
for (k in 2:6) {
  OR[k]          <- exp(beta[k])
}
# multivariate probit covariate imputation model
for (i in 1:n) {
  Z[i,1:2]        ~ dmnorm(mu[i,1:2],
                           Omega[1:2,1:2])I(lo[i,1:2],up[i,1:2])
  mu[i,1]        <- delta[1,1] + delta[2,1]*THM[i] +
                    delta[3,1]*male[i] + delta[4,1]*dep[i] +
                    delta[5,1]*area.smk[i] +
                    delta[6,1]*area.eth[i]
  mu[i,2]        <- delta[1,2] + delta[2,2]*THM[i] +
                    delta[3,2]*male[i] + delta[4,2]*dep[i] +
                    delta[5,2]*area.smk[i] +
                    delta[6,2]*area.eth[i]
}
for (i in 1:Nmis) { # Data file is ordered so subjects
                    # 1,...,Nmis have missing values
  smk[i]         <- step(Z[i,1]) # thresholded value of Z[i,1]
  eth[i]         <- step(Z[i,2]) # thresholded value of Z[i,2]
}
Sigma[1,1]       <- 1
Sigma[2,2]       <- 1
Sigma[1,2]       <- corr
Sigma[2,1]       <- corr
corr              ~ dunif(-1, 1)
Omega[1:2, 1:2] <- inverse(Sigma[,])
for (k in 1:6) {
  delta[k,1]     ~ dnorm(0, 0.0001)
  delta[k,2]     ~ dnorm(0, 0.0001)
}
```

```
node  mean    sd      MC error 2.5%    median  97.5%   start sample
OR[2] 1.178   0.121   0.002129 0.956   1.172   1.425   1001  20000
OR[3] 0.8095  0.08177 0.001518 0.6607  0.8052  0.9821  1001  20000
OR[4] 1.007   0.101   0.001919 0.8243  1.001   1.221   1001  20000
OR[5] 2.915   0.5202  0.01434  2.033   2.867   4.057   1001  20000
OR[6] 4.116   0.7181  0.01833  2.897   4.053   5.716   1001  20000
```

```
  corr  -0.3039 0.05583 0.002219 -0.4096 -0.3046 -0.1904 1001  20000
```

There is a small excess risk of low birthweight for mothers with high THM levels in their tap water supply, although the posterior 95% credible interval just includes the null odds ratio. Maternal smoking and non-white ethnicity confer substantially increased risks of low birthweight, although the wide credible intervals for these effects reflect uncertainty due to the high proportion of missing values. Analysis of the complete cases only produced a far more uncertain estimate of the THM effect (mean of `OR[2]` = 1.13, 95% CI 0.76 to 1.64), whilst analysis of the full data excluding the confounders `smk` and `eth` from the regression model produced an upwardly biased estimate of the THM effect (mean of `OR[2]` = 1.44, 95% CI 1.21 to 1.70).

9.2 Prediction

There are a number of reasons why we may want to predict an unknown quantity Y^{pred}. We may want to "fill in" missing or censored data (§9.1) or predict replicate datasets in order to check the adequacy of our model (§8.4). Finally, we may simply want to make predictions about the future.

If we were working within a classical paradigm, it would not be straightforward to make full predictions after fitting a statistical model. Although point predictions of a future quantity Y^{pred} may be easy, obtaining the appropriate full predictive distribution for Y^{pred} is challenging, as one needs to account for three components: uncertainty about the expected future value $E[Y^{\text{pred}}]$, the inevitable sampling variability of Y^{pred} around its expectation, and the uncertainty about the size of that error, as well as the correlations between these components. Fortunately, it is so trivial to obtain such predictive distributions using MCMC that it can be dealt with very briefly.

Suppose we have a model $p(y^{\text{pred}}|\theta)$ and a fully specified prior distribution $p(\theta)$. We have already seen in §2.7 how Monte Carlo methods can produce the predictive distribution of a future quantity as $p(y^{\text{pred}}) = \int p(y^{\text{pred}}|\theta)p(\theta)d\theta$ by simply including y^{pred} in the model and treating it as an unknown quantity. The same principle applies if instead we are using MCMC methods with a posterior distribution $p(\theta|y)$ (Chapter 4). In the example below we point out that, rather than explicitly include the quantities to be predicted in the model description, it may be easier to just expand the dataset to include missing data indicated as `NA`.

Example 9.2.1. *Dugongs (continued): prediction*
Consider again the growth model for dugongs from Example 6.3.1. Suppose we want to project the length of dugongs beyond the currently observed age range, say at ages 35 and 40 years. We could explicitly include, as quantities in the model, the *expected* lengths of all dugongs at those ages, as well as the *observable* lengths of specific future dugongs. Assuming

$$y_j \sim \text{Normal}(\mu_j, \sigma^2 = \tau^{-1}), \quad \mu_j = \alpha - \beta\gamma^{x_j}$$

for the observed data, we add the code

```
  mu35 <- alpha - beta*pow(gamma, 35)
  mu40 <- alpha - beta*pow(gamma, 40)
  y35   ~ dnorm(mu35, tau)
  y40   ~ dnorm(mu40, tau)
```

Alternatively, it will generally be easier to leave the model description unmodified and instead expand the data file with two dugongs of the appropriate ages but with missing lengths:

```
 list(N = 29,
      x = c(1.0, 1.5, 1.5, ..., 29.0, 31.5, 35, 40),
      y = c(1.80, 1.85, 1.87, ..., 2.72, 2.57, NA, NA))
```

Posterior predictive summaries for the quantities of interest are given by

```
  node   mean  sd      MC error 2.5%  median 97.5% start sample
  mu[28] 2.638 0.05949 0.002778 2.528 2.635 2.762 1001   10000
  mu[29] 2.642 0.06291 0.002959 2.528 2.638 2.775 1001   10000
  y[28]  2.637 0.1153  0.002654 2.408 2.638 2.865 1001   10000
  y[29]  2.642 0.1179  0.003107 2.413 2.64  2.881 1001   10000
```

The intervals around `mu[28]` and `mu[29]` reflect uncertainty concerning the fitted parameters α, β, and γ, as is the case for the other elements of `mu`. Intervals around the missing `y`s additionally reflect the sampling error σ and uncertainty about the value of σ. The model fit and predictive intervals for `mu[28]` and `mu[29]` are shown together in Figure 9.2. Widening of the intervals towards the right-hand side of the plot has nothing to do with the fact that the right-most intervals are predictive; this is simply due to greater uncertainty in the fitted curve for older animals.

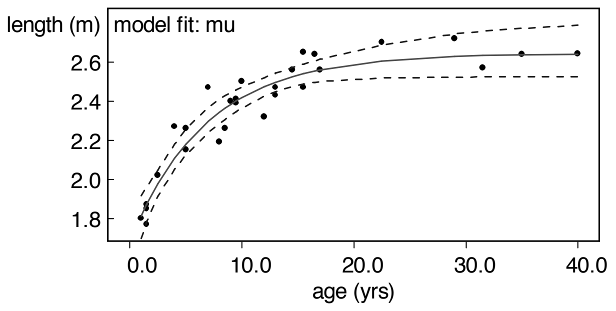

FIGURE 9.2
Model fit for observed dugongs data, with 95% posterior predictive intervals for the expected dugong length at ages 35 and 40 years. The points plotted (●) at 35 and 40 years are the posterior median values of `y[28]` and `y[29]`, which are specified as missing. The points coincide exactly with the predictive medians for `mu[28]` and `mu[29]`.

9.3 Measurement error

Errors in measurement can occur for both responses and covariates. The former case is straightforward, since standard statistical models can be thought of as encompassing errors in measurement and we can select the appropriate response distribution. When covariates are measured with error, there are two possible models: *classical* and *Berkson.*

The more common, classical model is represented in the graph shown in Figure 9.3(a), in which the observed covariate x is assumed conditionally independent of the response y given the "true" underlying covariate value z. The covariate may be categorical or continuous, but in either case needs to be provided with a prior distribution, with parameters ψ, say, which may or may not be known. In addition, an error model is assumed with parameters ϕ, which in order to be identifiable will need to be either assumed known or estimable from a subset of data in which both x and z are observed.

Example 9.3.1. *Cervix: case-control study with errors in covariates*
Carroll et al. (1993) consider the problem of estimating the odds ratio of a disease d in a case-control study where the binary exposure variable is measured with error. Their example concerns exposure to herpes simplex virus (HSV) in women with invasive cervical cancer ($d = 1$) and in controls ($d = 0$). Exposure to HSV is measured by a relatively inaccurate western blot procedure x for 1929 of the 2044 women, whilst for 115 women, it is also measured by a refined or "gold standard" method z. The data are given in Table 9.1. They show a substantial

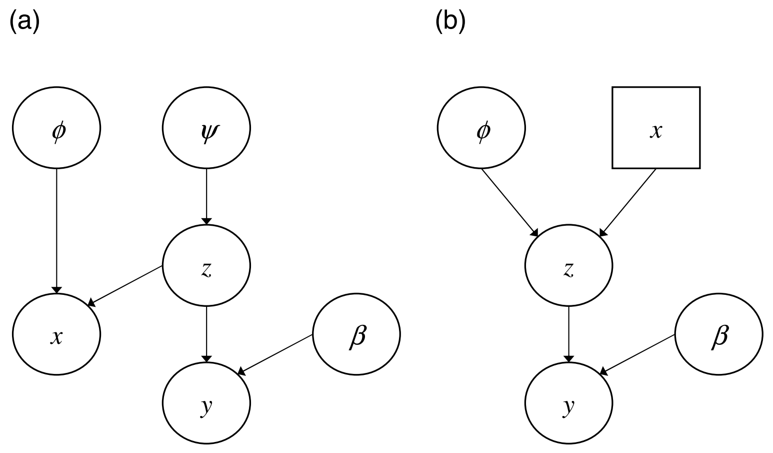

FIGURE 9.3
DAGs depicting classical and Berkson measurement error models. The response variable y is regressed on "true" covariates z, with regression coefficients β. (a) Classical model: the observed covariates x are assumed dependent on the true values z via an error model with parameters ϕ; a prior distribution with parameters ψ is specified for z. (b) Berkson error model: the true covariates z are assumed dependent on the observed values x via an error model with parameters ϕ.

amount of misclassification, as indicated by low sensitivity and specificity of x in the "complete" data. The degree of misclassification is also significantly higher for the controls than for the cases ($p = 0.049$ by Fisher's exact test).

A (prospective) logistic model is fitted to the case-control data as follows

$$d_i \sim \text{Bernoulli}(p_i), \quad \text{logit}(p_i) = \beta_0 + \beta z_i, \quad i = 1, \ldots, 2044,$$

where β is the log odds ratio of disease d. Since the relationship between d and z is only directly observable in the 115 women with "complete" data, and because there is evidence of differential measurement error, the following parameters are required in order to estimate the misclassification model:

$$\begin{aligned}
\phi_{11} &= \Pr(x = 1 | z = 0, d = 0) \\
\phi_{12} &= \Pr(x = 1 | z = 0, d = 1) \\
\phi_{21} &= \Pr(x = 1 | z = 1, d = 0) \\
\phi_{22} &= \Pr(x = 1 | z = 1, d = 1) \\
\psi &= \Pr(z = 1)
\end{aligned}$$

BUGS code for the model is as follows:

```
for (i in 1:n) {
```

TABLE 9.1
Case-control data for cervix example.

Complete data				Incomplete data			
d	z	x	Count	d	z	x	Count
1	0	0	13	1	—	0	318
1	0	1	3	1	—	1	375
1	1	0	5	0	—	0	701
1	1	1	18	0	—	1	535
0	0	0	33				
0	0	1	11				
0	1	0	16				
0	1	1	16				

```
    d[i]          ~ dbern(p[i])
    logit(p[i]) <- beta0 + beta*z[i]
    z[i]          ~ dbern(psi)
    x[i]          ~ dbern(phi[z1[i], d1[i]])
    z1[i]        <- z[i] + 1
    d1[i]        <- d[i] + 1
  }
  for (j in 1:2) {
    for (k in 1:2) {
      phi[j, k]   ~ dunif(0, 1)
    }
  }
  psi             ~ dunif(0, 1)
  beta0           ~ dnorm(0, 0.0001)
  beta            ~ dnorm(0, 0.0001)
```

where the `z1` and `d1` variables are created because `phi[]` must be indexed with 1s and 2s, as opposed to 0s and 1s, and functions are not allowed as indices. The data can be specified "long-hand" with three entries (`d`, `z`, `x`) for each of the 2044 individuals. Alternatively the individual-level data can be "constructed" from Table 9.1 via the following additional code:

```
  for (j in 1:8) {
    for (i in offset[j]:offset[j+1]-1) {
      d[i] <- d.com[j]; x[i] <- x.com[j]; z[i] <- z.com[j]
    }
  }
  for (j in 9:12) {
    for (i in offset[j]:offset[j+1]-1) {
      d[i] <- d.inc[j-8]; x[i] <- x.inc[j-8]
    }
  }
```

where `d.com`, `x.com`, and `z.com` denote the complete data, `d.inc` and `x.inc` denote the incomplete data, and `offset` contains the cumulative counts of individuals within each category: `offset=c(1, 14, 17, 22, 40, 73, 84, 100, 116, 817, 1352, 1670, 2045)`. Posterior summaries are given in the table below.

```
node     mean    sd      MC error 2.5%     median  97.5%   start sample
beta     0.6213  0.3617  0.01924  -0.09153 0.6188  1.345   1001  20000
beta0    -0.9059 0.199   0.01045  -1.321   -0.8996 -0.5283 1001  20000
phi[1,1] 0.3177  0.05309 0.00199  0.2109   0.3186  0.4199  1001  20000
phi[1,2] 0.2212  0.08055 0.003301 0.07556  0.2188  0.3884  1001  20000
phi[2,1] 0.5691  0.06352 0.002116 0.4428   0.5683  0.6941  1001  20000
phi[2,2] 0.7638  0.06187 0.002506 0.6409   0.7646  0.8806  1001  20000
psi      0.4923  0.04304 0.001771 0.4057   0.4929  0.5771  1001  20000
```

From this output we can estimate that the chance of falsely identifying HSV using a western blot is 32% in controls and 22% in cases, while the chance of missing a true HSV is 44% in controls and 24% in cases. Accounting for this misclassification results in a substantially de-attenuated estimate of the exposure log-odds ratio, although the increased uncertainty means that this is no longer statistically significant (posterior mean and 95% CI for `beta` = 0.62 (-0.09, 1.35) compared to 0.45 (0.27, 0.63) if covariate misclassification is ignored).

Example 9.3.2. *Dugongs (continued): measurement error on age*
Recalling again Example 6.3.1, we now assume that the observed age x_j is an imperfect measure of the true age z_j, with measurement standard deviation 1. We consider the model

$$y_j \sim \text{Normal}(\mu_j, \sigma^2), \quad \mu_j = \alpha - \beta\gamma^{z_j}, \quad x_j \sim \text{Normal}(z_j, 1),$$

with $\alpha, \beta \sim \text{Uniform}(0, 100)$, $\gamma \sim \text{Uniform}(0, 1)$, and $\log\sigma \sim \text{Uniform}(-10, 10)$. In addition, a prior distribution for each z_j is required. In the absence of prior knowledge we assume $z_j \sim \text{Uniform}(0, 100)$ for $j = 1, \ldots, n$. BUGS code for the model is given by

```
for(j in 1:n) {
  y[j]       ~ dnorm(mu[j], tau)
  mu[j]     <- alpha - beta*pow(gamma, z[j])
  x[j]       ~ dnorm(z[j], 1)
  z[j]       ~ dunif(0, 100)
}
alpha        ~ dunif(0, 100)
beta         ~ dunif(0, 100)
gamma        ~ dunif(0, 1)
tau         <- 1/sigma2
```

```
log(sigma2) <- 2*log.sigma
log.sigma    ~ dunif(-10, 10)
```

The model fit is shown in Figure 9.4. Note that the "true" ages z_j, $j = 1, \ldots, n$, are estimated such that the model fit is improved. This is reflected by a posterior median value for σ^2 of 0.0078, which is reduced from 0.0094 (when no measurement error was assumed — see Example 6.3.1). Figure 9.5 shows the posterior distribution of the difference between each observed and "true" age, calculated by adding the code:

```
for (j in 1:n) {resx[j] <- x[j] - z[j]}
```

Where the fit is not improved by estimating `z[j]` away from the observed value `x[j]`, the distribution of `resx[j]` is approximately standard normal. We can see that, particularly in the first half of the dataset, there is considerable adjustment of the ages being entered into the regression equation.

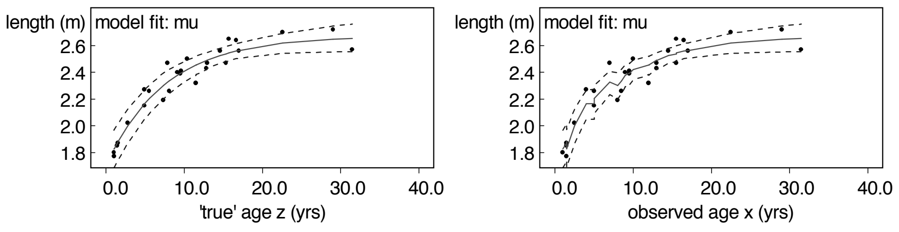

FIGURE 9.4
Model fits from analysis of dugongs data assuming a classical measurement error model for the observed ages. Left-hand side: model fit plotted against posterior median "true" ages, z_j, $j = 1, \ldots, n$. Right-hand side: model fit plotted against observed ages x_j, $j = 1, \ldots, n$.

Berkson errors arise in situations where the observed covariates are expected to be less variable than the "true" values, perhaps because the observed values are aggregated. This can occur when the covariates measure environmental exposure, say, such as levels of air pollution. The observed values may be summary measures for geographical areas, each, perhaps, taken at a single site or summed/averaged over the area. The *actual* exposures of individuals within those areas would then be expected to be more variable than the recorded values. This leads to a measurement error model in which the true covariate value depends on the observed value, rather than the other way round as in classical measurement error (see the right-hand side of Figure 9.3). The following example illustrates the use of a Berkson model for air pollution data.

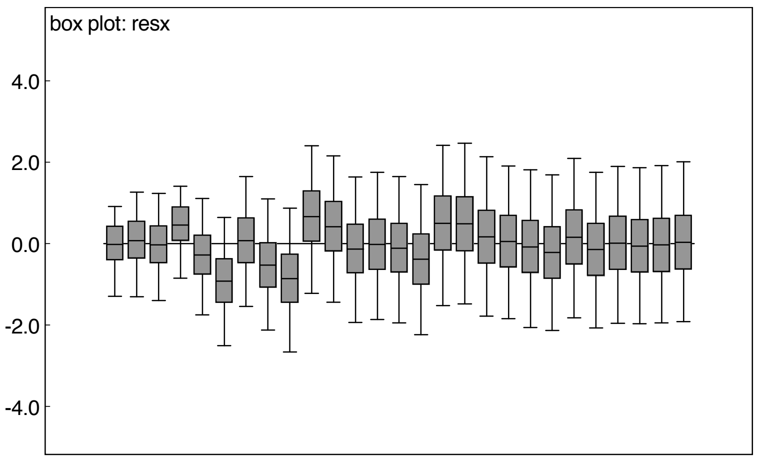

FIGURE 9.5
Box plot summarising posterior distributions of the difference between each observed and "true" age for the dugongs example with covariate measurement error.

Example 9.3.3. *Air pollution: Berkson measurement error*
Whittemore and Keller (1988) examine data regarding the potential effects of exposure to nitrogen dioxide (NO_2) on respiratory illness. One hundred and three children are categorised as having respiratory illness or not and of being exposed to one of three different levels of NO_2 in their bedrooms:

	Bedroom NO_2 level in ppb (x)			
Respiratory illness (y)	< 20	20–40	40+	Total
Yes	21	20	15	56
No	27	14	6	47
Total (n)	48	34	21	103

The three exposure categories (indexed by j) are thought of as a surrogate for "true exposure," and the nature of the measurement error relationship is known precisely from a separate calibration study:

$$z_j = \alpha + \beta x_j + \epsilon_j, \quad j = 1, 2, 3,$$

where x_j equals 10, 30, and 50 for $j = 1$, 2, and 3, respectively, and z_j is interpreted as the "true average value" of NO_2 in group j. In addition, from the calibration study we assume $\alpha = 4.48$, $\beta = 0.76$, and $\epsilon_j \sim \text{Normal}(0, 81.14)$, $j = 1, 2, 3$. We wish to fit the following logistic regression:

$$y_j \sim \text{Binomial}(p_j, n_j), \quad \text{logit}(p_j) = \theta_1 + \theta_2 z_j, \quad j = 1, 2, 3,$$

where n_j is the total number of children in group j.

```
for (j in 1:3) {
  y[j]          ~ dbin(p[j], n[j])
  logit(p[j]) <- theta[1] + theta[2]*z[j]
  z[j]          ~ dnorm(mu[j], 0.01232)
  mu[j]       <- alpha + beta*x[j]
}
theta[1]        ~ dnorm(0, 0.0001)
theta[2]        ~ dnorm(0, 0.0001)
```

```
node     mean    sd      MC error 2.5%      median  97.5%  start sample
theta[1] -0.8096 0.8559  0.03736  -2.889    -0.6557 0.3502 12001 20000
theta[2] 0.04207 0.03144 0.001313 -0.002226 0.03667 0.1212 12001 20000
z[1]     12.8    8.299   0.2011   -3.881    12.98   28.81  12001 20000
z[2]     27.43   7.474   0.08438  12.95     27.37   42.39  12001 20000
z[3]     41.43   8.56    0.1437   25.35     41.23   58.56  12001 20000
```

Note that the effect of NO_2 exposure on the chances of developing a respiratory illness is almost, but not quite, significant in this analysis: the 95% credible interval for θ_2, which represents the log odds ratio for a unit increase in "true exposure" only just includes zero.

9.4 Cutting feedback

In the dugongs example 9.3.2 above the true ages are estimated such that they improve the fit of the line. In many cases, this is exactly what we would want: the information in the measured ages regarding the values of the true ages is supplemented by feedback from the response data (dugong lengths) due to the assumed relationship between length and age. However, there are situations in which we might wish to infer the values of missing variables based solely on the observed values of those variables. In other words, we may wish to ignore, or "cut" the feedback from the response data. We can achieve this using the `cut()` function in WinBUGS,[*] as illustrated in the following example.

Example 9.4.1. *Cutting feedback*
Consider the simple linear regression presented in Figure 9.6(a). In this case the values of the variable plotted on the x-axis are assumed known. However, suppose we know that they are measured with error and that the standard deviation of those errors is 1.5. We denote the response variable by y_i, $i = 1, \ldots, n$, and

[*]or OpenBUGS. There is no "cut" function currently in JAGS.

the modelled and observed values of the independent variable by z_i and x_i, $i = 1, \ldots, n$, respectively. (Note that $x_i = i$, $i = 1, \ldots, n$.) One option is to assume:

$$y_i \sim \text{Normal}(\mu_i, \sigma^2), \quad \mu_i = a + bz_i, \quad x_i \sim \text{Normal}(z_i, 1.5^2),$$

with appropriate priors on a, b, σ, and each z_i. This would allow estimation of the z_is to be influenced by feedback from the y_is. To cut this feedback, we assume instead

$$z_i = \text{cut}(z_i^*), \quad x_i \sim \text{Normal}(z_i^*, 1.5^2),$$

with appropriate priors on the z_i^*s, e.g., $z_i^* \sim \text{Uniform}(-100, 100)$, $i = 1, \ldots, n$. The $\text{cut}(.)$ function here makes a copy of the variable passed as an argument but otherwise severs the link between argument and result, z_i^* and z_i, respectively, in this case. Hence z_i always has the same value as z_i^* but z_i^* is isolated from the y_is and cannot be influenced by them. The following BUGS code fits models with and without feedback as well as the model in which $z_i = x_i$, $i = 1, \ldots, n$. In order to fit multiple models simultaneously we must make multiple copies of the dataset $\{y_i, x_i, i = 1, \ldots, n\}$, as shown below.

```
model {
  for (m in 1:3) {
    for (i in 1:n) {
      y.copy[m, i] <- y[i]
      x.copy[m, i] <- x[i]
      y.copy[m, i]  ~ dnorm(mu[m, i], tau[m])
      mu[m, i]     <- a[m] + b[m]*z[m, i]
    }
    a[m]            ~ dnorm(0, 0.0001)
    b[m]            ~ dnorm(0, 0.0001)
    tau[m]         <- 1/pow(sigma[m], 2)
    sigma[m]        ~ dunif(0, 100)
  }
  for (i in 1:n) {
    z[1, i]        <- x.copy[1, i]
    x.copy[2, i]    ~ dnorm(z[2, i], 0.4444)
    z[2, i]         ~ dunif(-100, 100)
    z[3, i]        <- cut(z.star[i])
    x.copy[3, i]    ~ dnorm(z.star[i], 0.4444)
    z.star[i]       ~ dunif(-100, 100)
  }
}
```

Model fits for the models with and without feedback are shown in Figure 9.6(b) and Figure 9.6(c), respectively. Note that with feedback the z_is are estimated so that the regression line fits as well as possible. One danger of allowing this to happen is that the estimates may become implausible. Indeed, note that in this example several of the z_is are not ordered when feedback is allowed, e.g., posterior median estimates for z_3, z_4, and z_5 are 2.90, 5.59, and 3.40, respectively, although there is considerable overlap between the 95% credible intervals: (0.844, 4.81), (2.85, 7.10), and (1.70, 5.98), respectively. It may be known that they must be ordered, however. An ordering constraint could be applied in such cases, as discussed in § 9.7.2, but there are numerous situations in which an obvious constraint to ensure plausibility does not exist. The reader is referred to Lunn et al. (2009a) for further discussion on difficulties with feedback. Without feedback, point estimates for the z_is are approximately equal to the measured values, $x_i = i$, $i = 1, \ldots, n$, but considerable uncertainty is acknowledged and this is propagated into the regression analysis, manifesting as a wider credible interval for the model fit compared to when the observed x_is are assumed error-free.

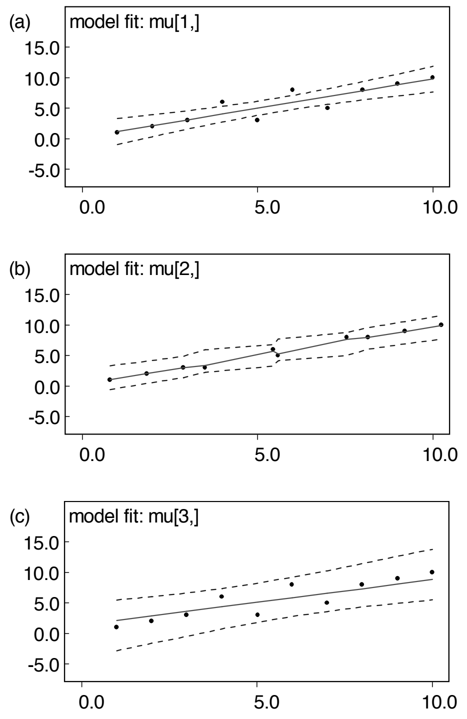

FIGURE 9.6
Model fits for linear regression data, under various assumptions for the independent variable: (a) x_is assumed error-free, i.e., $z_i = x_i$; (b) fully Bayesian model (with feedback); and (c) model without feedback. In (b) and (c), the model fits are plotted against posterior mean estimates of the z_is.

9.5 New distributions

9.5.1 Specifying a new sampling distribution

Suppose we wish to use a sampling distribution that is not included in the list of standard distributions, in which an observation y_i contributes a likelihood term L_i. One possibility is the "zeros trick" based on the following.

We invent a set of observations $z_i = 0$, each of which is assumed to be drawn from a Poisson(ϕ_i) distribution. Each then has a likelihood contribution $\exp(-\phi_i)$, and so if ϕ_i is set to $-\log(L_i)$, we will obtain the correct likelihood contribution. (Note that ϕ_i should always be > 0 as it is a Poisson mean, and so we may need to add a suitable constant to ensure that it is positive.) The BUGS code will look like the following:

```
const    <- 10000   # arbitrary, ensures phi[i] > 0
for (i in 1:n) {
  z[i]   <- 0
  z[i]    ~ dpois(phi[i])
  phi[i] <- -log(L[i]) + const
  L[i]   <- ...
}
```

L_i is set to a function of y_i and θ proportional to the likelihood $p(y_i|\theta)$. This trick allows arbitrary sampling distributions to be used and is particularly suitable when, say, dealing with truncated distributions (§9.6.2).

A new observation from the distribution, denoted y_{n+1}, can be predicted by including it as an additional, but missing, observation in the data file and assigning it an improper uniform prior, e.g., `y[n+1]` $\sim$ `dflat()`, defining z_i and ϕ_i in the same way as before for $i = n + 1$. The missing observation is essentially assumed to be an unknown parameter with a uniform prior, but also with a likelihood term corresponding to the sampling distribution.

Note that the DIC (§8.6) for data from distributions specified using the zeros trick, as reported by the WinBUGS or OpenBUGS DIC tool, is calculated with respect to z_i, not y_i. Example 11.6.2 explains how to transform this to the scale of y_i, so it can be compared with the DICs of models for y_i which are specified using built-in sampling distributions.

Example 9.5.1. *A clumsy way of modelling the normal distribution*
We use the "zeros trick" to model a normal distribution with unknown mean μ and unknown standard deviation σ, including predicting a new observation. We

have seven observed values and one missing value as follows: `y = c(-1, -0.3, 0.1, 0.2, 0.7, 1.2, 1.7, NA)`.

```
  for (i in 1:8) {
    z[i]   <- 0
    z[i]    ~ dpois(phi[i])
    phi[i] <- log(sigma) + 0.5*pow((y[i] - mu)/sigma, 2)
  }
  y[8]      ~ dflat()
  sigma     ~ dunif(0, 100)
  mu        ~ dunif(-100, 100)
```

We must provide an initial value for `y[8]`, via `y = c(NA, NA, NA, NA, NA, NA, NA, 0)`, say, otherwise BUGS will try to generate one from the improper prior and crash.

```
node  mean   sd     MC error 2.5%    median 97.5% start sample
mu    0.365  0.4758 0.006864 -0.5948 0.3693 1.316 4001  10000
sigma 1.18   0.481  0.01139  0.6216  1.067  2.415 4001  10000
y[8]  0.3499 1.355  0.03415  -2.345  0.3564 3.095 4001  10000
```

Whilst the results match those that would be obtained in a standard analysis using `y[i] ~ dnorm(mu, tau); tau <- 1/pow(sigma,2)`, this is an inefficient procedure, particularly for the prediction, and so a long run is necessary. The MC error for the prediction is 0.03 using the zeros trick and 0.01 for the same number of iterations in the equivalent standard analysis.

An alternative to the "zeros trick" is the "ones trick." Here we invent a set of observations equal to 1 instead, and assume each to be Bernoulli distributed with probability p_i. By making each p_i proportional to L_i (i.e., by specifying a scaling constant large enough to ensure $p_i < 1$ for all i), the required likelihood term is provided:

```
const  <- 10000   # arbitrary, ensures p[i] < 1
for (i in 1:n) {
  z[i] <- 1
  z[i]  ~ dbern(p[i])
  p[i] <- L[i]/const
}
```

We will illustrate use of the "ones trick" in §9.6.2, where we consider how to specify truncated sampling distributions.

9.5.2 Specifying a new prior distribution

Suppose we want to use a prior distribution for θ that does not belong to the standard set. Then we can use the "zeros trick" (see above) at the prior level combined with an improper uniform prior for θ. A single Poisson observation equal to zero, with mean $\phi = -\log(p(\theta))$, contributes a term $\exp(-\phi) = p(\theta)$ to the likelihood for θ; when this is combined with a "flat" prior for θ the correct prior distribution results. This is essentially the same process as predicting from a new distribution covered in the previous section. Summary BUGS code is

```
z    <- 0
z     ~ dpois(phi)
theta ~ dflat()
phi  <- expression for -log(desired prior for theta)
```

For example, if we wished to produce a standard normal prior, we would use

```
phi  <- 0.5*pow(theta, 2)
```

It is important to note that this method produces high auto-correlation, poor convergence, and large MC errors, so it is computationally slow and long runs are necessary. Initial values also need to be specified as the `dflat()` prior cannot be sampled from using `gen.inits`.

New sampling distributions and new prior distributions can also be specified in WinBUGS via the WinBUGS Development Interface (WBDev). This can give big computational savings and clearer BUGS code, at the cost of "lower-level" programming in Component Pascal — see § 12.4.8 for more details. There are similar but less well-documented capabilities in OpenBUGS and JAGS; see Chapter 12.

9.6 Censored, truncated, and grouped observations

9.6.1 Censored observations

A data point is a censored observation when we do not know its exact value, but we do know that it lies above or below a point c, say, or within a specified interval. The most common application is in survival analysis (§11.1), but here we consider general measurement problems. There are two strategies within BUGS:

1. In general, in WinBUGS we can use the `I(,)` construct (§A.2.2), which specifies a restricted range within which the unknown quantity lies. The unknown quantity is then simply treated as a model parameter. Note that in OpenBUGS the `C()` function is preferred (see §12.5.1) and JAGS uses a different syntax altogether (see §12.6.2).

2. Each exact observation y contributes $p(y|\theta)$ to the likelihood of θ, whereas an observation censored at c provides a contribution of $\Pr(Y >$

$c|\theta)$ or $\Pr(Y < c|\theta)$. Hence, if the distribution function can be expressed in BUGS syntax, then we can use either the "ones trick" or the "zeros trick" (§9.5) to directly specify the contribution of the censored observations to the likelihood.

Example 9.6.1. *Censored chickens*
Suppose we weigh nine chickens, with a scale that only goes up to 8 units, so that if the scale shows 8 it means that the chicken weighs *at least* 8 units, which we denote 8+. The weights are 6, 6, 6, 7, 7, 7, 8+, 8+, 8+. The population of chickens is assumed to have weights that are normally distributed with mean μ and standard deviation 1 unit. (This is not intended to be a realistic example — all the observed weights are integer-valued and would more realistically be modelled as roundings of a true continuous-valued weight, as in Example 9.6.3). If the 8+ weighings were exactly 8, and μ was assigned a locally uniform prior, then the posterior distribution for μ would be Normal$(7, 1/9)$. WinBUGS code accounting for the censoring via method 1 above is

```
model {
  for (i in 1:6) {y[i] ~ dnorm(mu, 1)}        # uncensored data
  for (i in 7:9) {y[i] ~ dnorm(mu, 1)I(8,)}   # censored data
  mu                   ~ dunif(0, 100)
}
data:
  list(y = c(6,6,6,7,7,7,NA,NA,NA))

  node mean  sd     MC error 2.5%  median 97.5% start sample
  mu   7.193 0.3478 0.003604 6.515 7.19   7.875 1001  10000
  y[7] 8.571 0.4809 0.005356 8.018 8.446  9.805 1001  10000
```

We note that the posterior mean of μ is greater than 7 since the censored observations have been estimated to be between 8.0 and 9.8 — see Figure 9.7 for the posterior density of censored observation `y[7]` (`y[8]` and `y[9]` also have the same posterior). The posterior standard deviation of μ is 0.35, slightly greater than if the data had been exact rather than censored — we can think of the effective sample size having been reduced from 9 to $1/0.3478^2 = 8.3$.

Now consider the second method outlined above, making use of the "zeros trick." Each censored observation provides a term $\Pr(Y > 8|\mu)$ to the likelihood of μ, which is equal to $\Pr(Y - \mu > 8 - \mu) = 1 - \Phi(8 - \mu) = \Phi(\mu - 8)$, where $\Phi(.)$ is the cumulative distribution function of the standard normal distribution, available in BUGS via the syntax `phi(.)`:

```
  for (i in 1:6) {y[i] ~ dnorm(mu, 1)}
  for (i in 1:3) {
    zeros[i]         <- 0
    zeros[i]          ~ dpois(p[i])
```

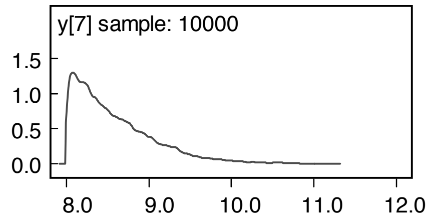

FIGURE 9.7
Posterior distribution for censored observation y[7] in the "censored chickens" example.

```
    p[i]              <- -log(phi(mu-8))
  }
  mu                   ~ dunif(0, 100)

  node mean  sd    MC error 2.5%  median 97.5% start sample
  mu   7.192 0.348 0.003687 6.519 7.185  7.882 1001  10000
```

We obtain largely the same results as with method 1, with similar Monte Carlo standard errors. Method 2 is computationally more efficient, as the censored observations are integrated out before analysis. However, method 1 is more generally applicable, as it does not require the distribution function to be known.

9.6.2 Truncated sampling distributions

A sampling distribution is truncated if for some reason we never observe cases above or below a specified point, although in the permissible range of observations the data follow a standard distribution. The sampling distribution must therefore be normalised to condition on lying in the permissible range, say $Y < c$, so that the likelihood contribution of an observation y is $p(y|\theta)/\Pr(Y < c|\theta)$. This will not generally be of standard form, and so either a new distribution has to be defined (§12.4.8) or the "ones"/"zeros" trick used (§9.5).

It is very important to realise that the I(,) construct is not appropriate for truncated distributions with unknown parameters, since the generated likelihood term will ignore the truncation and be incorrect (see Appendix A.2.2). However, the I(,) construct can be used when specifying truncated prior distributions with no unknown parameters — see Examples 5.3.2 and 6.3.1.

Example 9.6.2. *Truncated chickens*
In Example 9.6.1, suppose that any chicken weighing 8 or more units is sent back to get more exercise, so the distribution of chicken weights is right-truncated at

8. Therefore we only hear about the six chickens that weighed 6 or 7 units, and each of these provides a likelihood contribution of $\exp[-(y_i - \mu)^2/2]/\Phi(c - \mu)$. So using the "ones trick":

```
for (i in 1:6) {
  z[i] <- 1
  z[i]  ~ dbern(p[i])
  p[i] <- exp(-0.5*(y[i] - mu)*(y[i] - mu))/phi(8 - mu)
}
mu      ~ dunif(0, 100)

node mean  sd     MC error 2.5%  median 97.5% start sample
mu   6.737 0.4965 0.00498  5.819 6.72   7.75  1001  10000
```

Although we don't know how many chickens were weighed and returned, knowledge of this truncation has raised the estimated population mean from the sample mean of 6.5 to 6.74. The posterior standard deviation of 0.5 means that the effective sample size is $1/0.5^2 = 4$, so the truncation has considerably reduced the precision. Note that in this situation chickens weighing 8 or more units are not included in the data collection process, whereas in the censoring example (Example 9.6.1), while they may not be fully observed, they may still be present.

In JAGS, the `T(,)` construct may simply be used to truncate the sampling distribution in the following way. JAGS computes the appropriate normalising constant internally.

```
y[i] ~ dnorm(mu, 1)T(,8)
```

A similar general facility is planned in OpenBUGS but is only partially implemented currently.

As may be apparent from the examples above, there are subtle differences between truncation and censoring. Truncation is appropriate when values outside a given range are actually impossible. Censoring, on the other hand, is appropriate when values beyond that range are possible *in principle*, but have not been observed due to the nature of the measurement device/method — that is, they may be observable using a different method.

9.6.3 Grouped, rounded, or interval-censored data

If observations are either grouped into categories or rounded, to the nearest integer, say, the information provided by an observation is that it lies in a particular interval, say $(lower, upper)$. This can be treated as interval censoring and handled by assuming we have a true but unobserved quantity z which contributes to the likelihood for θ but we only know it lies between *lower* and *upper*. This is specified using `I(lower,upper)`.

Example 9.6.3. *Grouped chickens*
Suppose all nine chickens in Example 9.6.1 have been weighed and reported as 6, 6, 6, 7, 7, 7, 8, 8, 8, but we know that when the scales report 7 units, say, the true weight z could be anything between 6.5 and 7.5.

```
for (i in 1:9) {
  lower[i] <- y[i] - 0.5
  upper[i] <- y[i] + 0.5
  z[i]      ~ dnorm(mu, 1)I(lower[i], upper[i])
}
mu          ~ dunif(0, 100)

node mean  sd     MC error 2.5%  median 97.5% start sample
mu   7.001 0.3496 0.00364  6.322 7.003  7.692 1001  10000
z[1] 6.08  0.2778 0.002909 5.543 6.113  6.483 1001  10000
z[4] 6.993 0.2829 0.002573 6.527 6.991  7.474 1001  10000
z[7] 7.922 0.2759 0.002482 7.517 7.885  8.458 1001  10000
```

The posterior mean is 7, as might be expected from symmetric data, but the effective sample size is reduced from 9 to $1/0.35^2 = 8.2$ by the grouping. The true weight for a chicken reported as 6 is estimated to be 6.08, slightly "shrunk" towards the mean by the assumption that the weights in the population are normally distributed — see Figure 9.8.

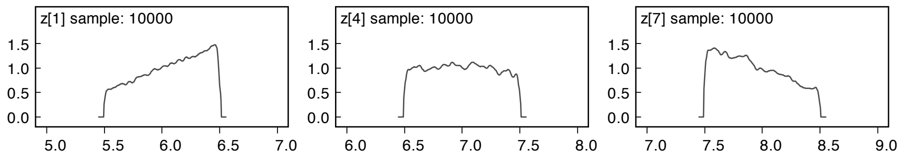

FIGURE 9.8
Posterior density estimates for "true" chicken weights in the "grouped chickens" example.

9.7 Constrained parameters

9.7.1 Univariate fully specified prior distributions

A single parameter θ may be subject to a range constraint such as $\theta > 0$. Provided the distribution of θ does not contain any unknown parameters, then this can be accommodated by using the `I(,)` construct (§9.6). For example, a standard normal variable constrained to be positive is expressed as

```
theta ~ dnorm(0,1)I(0,)
```

See Examples 5.3.2 and 6.3.1. An alternative approach, which better generalises to more complex constraints, is essentially an extension of the "ones trick." We introduce an auxiliary observation z taking the value 1. This is assumed to arise from a Bernoulli distribution whose parameter takes the value 1 if the constraint is obeyed, and 0 otherwise. When sampling θ, only values that obey the constraint, and therefore provide non-zero likelihood, will be accepted, as illustrated in the example below.

Example 9.7.1. *Half-normal*
The code below shows the half-normal distribution being generated in two different ways.

```
theta[1]    ~ dnorm(0,1)I(0,)
theta[2]    ~ dnorm(0,1)
z          <- 1
z           ~ dbern(constraint)
constraint <- step(theta[2])
```

The results show the substantially increased Monte Carlo error associated with the auxiliary data method:

```
node     mean   sd     MC error 2.5%    median 97.5% start sample
theta[1] 0.8009 0.6106 0.006119 0.03056 0.6696 2.253 4001  10000
theta[2] 0.7897 0.5973 0.01643  0.02286 0.6743 2.206 4001  10000
```

9.7.2 Multivariate fully specified prior distributions

Order constraints on a series of parameters can be expressed using the `I(,)` construct, provided the prior distribution does not contain unknown parameters. For example, to order `a[1]<a[2]<a[3]`:

```
a[1] ~ dnorm(0, 0.001)I(, a[2])
a[2] ~ dnorm(0, 0.001)I(a[1], a[3])
a[3] ~ dnorm(0, 0.001)I(a[2], )
```

Or as in Example 7.3.1, `a[2]` and `a[3]` could be defined by adding positively distributed increments to `a[1]` and `a[2]`, respectively. In JAGS and OpenBUGS, the elements of an unconstrained vector can also be sorted using the `sort()` function, for example, `b[1:3] <- sort(a[])`.

The auxiliary data method can be used to impose more complex constraints, as the following example shows.

Example 9.7.2. *Doughnut: bivariate normal with a hole in it*
Suppose we assume $\theta_i \sim$ Normal$(0, 1)$, $i = 1, 2$, but with the curious constraint that $\theta_1^2 + \theta_2^2 > 1$. This can be generated using the following code.

```
theta[1]     ~ dnorm(0, 1)
theta[2]     ~ dnorm(0, 1)
z           <- 1
z            ~ dbern(constraint)
constraint <- step(theta[1]*theta[1] + theta[2]*theta[2] - 1)
```

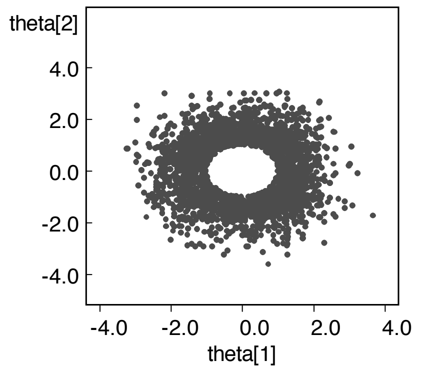

FIGURE 9.9
Scatterplot showing 5000 samples from a bivariate normal distribution subject to the constraint $\theta_1^2 + \theta_2^2 > 1$.

A scatterplot of 5000 simulations (Figure 9.9) shows a bivariate normal distribution with a hole in the centre.

Additive constraints on parameters can be easily imposed by reparameterisation. For example, if we require a set of parameters β_i, $i = 1, \ldots, n$, to sum to 0, we can define each as $\beta_i = b_i - \overline{b}$, where the b_is are independent with known prior distributions.

Example 9.7.3. *Bristol (continued): sum-to-zero constraint*
We consider Example 8.3.1 as a logistic model:

$$y_i \sim \text{Binomial}(\theta_i, n_i), \quad \text{logit}\, \theta_i = \alpha + \beta_i, \quad i = 1, \ldots, 12.$$

We allow each centre i to have its own effect parameter β_i, but with the commonly imposed constraint that those parameters add to 0, in order to ensure identifiability.

```
for (i in 1:12) {
  y[i]              ~ dbin(theta[i], n[i])
  logit(theta[i]) <- alpha + beta[i]
  beta[i]         <- b[i] - mean(b[])
  b[i]              ~ dunif(-10,10)
}
alpha               ~ dunif(-10,10)
```

Figure 9.10 shows a box plot of the `beta[i]`s from the above model. These are identifiable, due to the sum-to-zero constraint, but the individual `b[i]`s are not. Non-identifiable parameters can actually be introduced to improve convergence in hierarchical models — see §10.5.

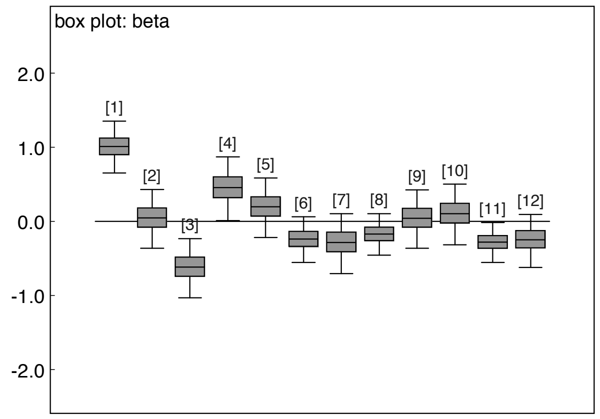

FIGURE 9.10
Posterior summaries for centre effects `beta[i]` in the Bristol example.

9.7.3 Prior distributions with unknown parameters

All of the parameter constraints considered thus far have taken a standard prior distribution and truncated it. Those standard priors have also had fixed parameters. Unfortunately, when the prior parameters are unknown, the `I()` method for truncation does not work correctly in BUGS — it is simply a trick for restricting the values sampled for a given node and only works in simple settings. In particular, the normalising constant for the truncation is not accounted for when computing the likelihood contribution of the truncated parameter(s) to the full conditional distribution(s) of the prior parameter(s). In such cases, we have several alternative options, but as this issue is particularly relevant to hierarchical models, discussion of these is deferred until §10.2.2.

9.8 Bootstrapping

As suggested in Chapter 2, BUGS can, in principle, be used to perform any statistical procedure based on random sampling, which need not necessarily be a Bayesian analysis. For example, we can implement classical nonparametric bootstrap estimation as follows.

Example 9.8.1. *Bootstrapping in BUGS: the Newcomb data*
The Newcomb data have previously been considered as an illustration of model elaboration for non-normal data (see Examples 8.2.1, 8.4.3, and 8.4.4). Carlin and Louis (2008) point out that, given the outliers, a more robust analysis might consider inference on the median rather than the mean of the distribution. If we wish to avoid a parametric assumption about the shape of the distribution, then we can adopt the basic bootstrap procedure of taking a series of repeat samples with replacement and calculating the sample mean and median for each of these repeats. This can be easily carried out in BUGS using the code below.

```
for (i in 1:N) {
  p[i]     <- 1/N          # set up uniform prior on 1 to N
}
for (j in 1:N){
  pick[j]   ~ dcat(p[])    # pick random number between 1 and N
  Yboot[j] <- Y[pick[j]]   # set jth bootstrap observation
}
mean       <- mean(Yboot[])
# now find median of bootstrap sample: this is halfway
# between observation N/2 and N/2+1...
n1         <- N/2
```

```
n2        <- n1 + 1
median    <- (ranked(Yboot[], n1) + ranked(Yboot[], n2))/2
```

We note how the discrete uniform prior distribution is set up and the use of the `dcat` distribution to select a random observation. We monitor one of the bootstrap elements `Yboot[1]`; its density in Figure 9.11 provides an approximate sampling distribution for the data. The median has a discrete distribution with median 27 and 95% interval 26.0 to 28.5 (the true value for the speed of light would give 33, well outside the 95% bootstrap interval).

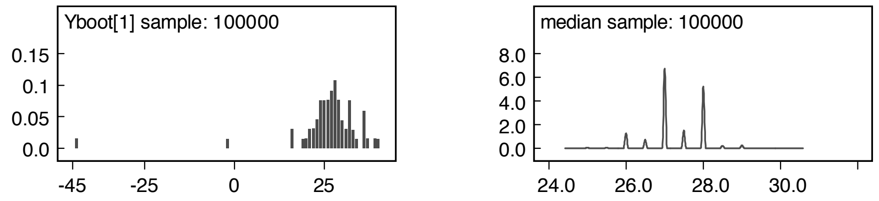

FIGURE 9.11
Empirical distributions for `Yboot[1]` and `median` based on 100,000 simulations.

9.9 Ranking

There is increasing attention to the profiling of schools, hospitals, and so on, often resulting in institutions being ranked into a league table similar to sports teams or competitors. Generally the rank is treated as a descriptive statistic, but we can also think of the observed rank as an imperfect measure of the "true rank" and perform statistical inference. This can be useful when we want to assess the probability, for example, that a treatment that currently looks best is truly the best treatment being examined.

The observed rank is a highly unreliable summary statistic since it can be very sensitive to small changes in the data. Bayesian methods can provide posterior interval estimates for ranks for which WinBUGS and OpenBUGS contain "built-in" options.[†] `rank(x[], i)` returns the rank of the i^{th} element

[†]In JAGS, `rank(x[])` transforms a vector `x` into a vector of ranks, so that the equivalent of `rank(x[],i)` is `y <- rank(x[]); y[i]`. The equivalent of `ranked(x[], i)` is `y <- sort(x[]); y[i]`.

of `x`. `equals(rank(x[],i),1)` $= 1$ if the i^{th} element of `x` has the lowest value, and 0 otherwise; the mean is the probability that the i^{th} element has the lowest value. `ranked(x[], i)` returns the value of the i^{th}-ranked element of `x`. The `Rank` option of the `Inference` menu monitors the rank of each element of a specified vector.

Example 9.9.1. *Bristol (continued): ranking*
Consider again the child heart surgery mortality rates introduced in Example 8.3.1. Ignoring Bristol, we consider whether the variability in mortality rates between hospitals allows any confident ranking. We assume the mortality in each hospital is $p_i, i = 1, ..., N$, with $N = 11$, which are assumed to have independent Jeffreys Beta(0.5,0.5) priors. We would like to assess the true rank of each hospital and the probability that each has the highest or lowest mortality.

```
for (i in 1:N) {
  numbers1toN[i]  <- i
  p[i]             ~ dbeta(0.5, 0.5)
  r[i]             ~ dbin(p[i], n[i])
  hosp.rank[i]    <- rank(p[], i)            # rank of hospital i
  prob.lowest[i]  <- equals(hosp.rank[i], 1) # =1 if hosp i is lowest
  prob.highest[i] <- equals(hosp.rank[i], N) # =1 if hosp i is highest
}
hosp.lowest       <- inprod(numbers1toN[], prob.lowest[])
                                             # index of lowest hosp
hosp.highest      <- inprod(numbers1toN[], prob.highest[])
                                             # index of highest hosp
```

The `rank` function produces the rank of each hospital at each iteration so that, for example, `hosp.rank[i]` $= 1$ if hospital `i` currently has the lowest mortality rate `p[i]`. `prob.lowest[i]` will then be 1 at that iteration, and so the mean of `prob.lowest[i]` will provide the probability that hospital `i` is the "safest" hospital.

We note the "which is min/max" trick used to pick out the index of, say, the lowest hospital: `prob.lowest[]` is a vector of zeros except for a 1 in the position of the hospital with the lowest `p` at that iteration; by taking an inner-product $\sum_{\texttt{i}}$ `i*prob.lowest[i]`, `hosp.lowest` takes on the value equal to the index of the hospital currently ranked lowest.

The upper panel of Figure 9.12 shows that there are substantial posterior probabilities (0.71 and 0.68, respectively) of hospital 2 being the "safest" (with the lowest mortality) and hospital 3 being the "least safe": these are hospitals 3 and 4 in the original data table in Example 8.3.1. The lower panel illustrates that there is considerable uncertainty regarding each hospital's rank, however.

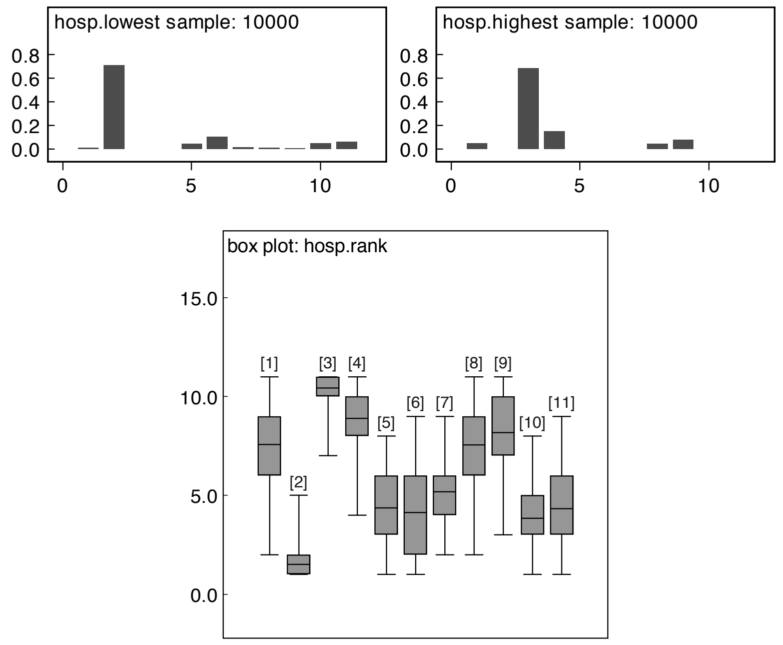

FIGURE 9.12
Top: Posterior histograms for the hospital with the lowest (left) and highest (right) mortality. Bottom: Box plot comparing hospital-specific posterior ranks.