CHAPTER 19

Debugging and speeding convergence

Once data and a model have been set up, we face the challenge of debugging or, more generally, building confidence in the model and estimation. The steps of Bugs and R as we have described them are straightforward, but cumulatively they require a bit of effort, both in setting up the model and checking it—adding many lines of code produces many opportunities for typos and confusion. In Section 19.1 we discuss some specific issues in Bugs and general strategies for debugging and confidence building. Another problem that often arises is computational speed, and in Sections 19.2–19.5 we discuss several specific methods to get reliable inferences faster when fitting multilevel models. The chapter concludes with Section 19.6, which is not about computation at all, but rather is a discussion of prior distributions for variance parameters. The section is included here because it discusses models that were inspired by the computational idea described in Section 19.5. It thus illustrates the interplay between computation and modeling which has often been so helpful in multilevel data analysis.

19.1 Debugging and confidence building

Our general approach to finding problems in statistical modeling software is to get various crude models (for example, complete pooling and no pooling, or models with no predictors) to work and then to gradually build up to the model we want to fit. If you set up a complicated model and you can't get it to run—or it will run but its results don't make sense—then either build it from scratch, or strip it down until you can get it to work and make sense. Figure 19.1 illustrates.

It might end up that there was a bug in your original data and model specification, or maybe the model you wanted to fit is not appropriate for your data, and you will move to a more reasonable version. Thus, the debugging often serves a statistical goal too, by motivating the exploration of various approximate and alternative models.

Getting your Bugs program to work

In R, C, and other command-based languages, if a function or script does not work, we have various direct debugging tactics, including executing the code line by line to find where it fails, executing parts of the code separately, and inserting print statements inside the code to display intermediate results. None of these methods work with Bugs, because the lines of a Bugs model are not executed sequentially. Actually, the lines of a Bugs model are not "executed" at all; rather, Bugs parses the entire model and then runs a process.

If there is a problem with the model, or the data, or the initial values, or the simulation itself, Bugs will crash (usually by halting and displaying an error message in the Bugs window—which you will see if you set the `debug=TRUE` option when calling from R—or occasionally by simply not responding, in which case you must close the Bugs window to end the process.

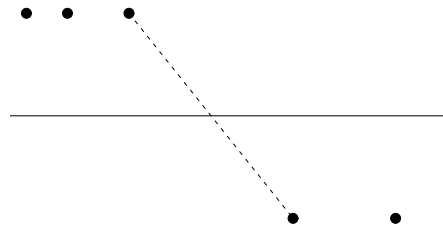

Figure 19.1 *Diagram of advice for debugging. The asterisk on the lower right represents the scenario in which problems arise when trying to fit the desired complex model. The dots on the upper left represent successes at fitting various simple versions, and the dots on the lower right represent failures at fitting various simplifications of the full model. The dotted line represents the idea that the problems can be identified somewhere between the simple models that fit and the complex models that don't.*

Thus, when Bugs fails, we can try to identify the problem from the error message in the Bugs window (as discussed below), but the only general approach is to go back to simpler models that work and then locate the error as indicated in Figure 19.1. Here we briefly list some specific problems that we have encountered in Bugs modeling.

Model not compiling. Common problems in the compilation include: too much inside a distribution argument (for example, `y[i]~dnorm(a+b*x[i],tau)` should be two lines: `y[i]~dnorm(y.hat[i],tau)` and `y.hat[i]<-a+b*x[i]`; see page 354); space in the wrong place in an assignment statement (for example, the expression `tau <- pow (sigma,-2)` should not have a space after "`pow`"); undefined parameters (that is, parameters used somewhere in the Bugs model but not modeled (with "∼") or assigned (with "`<-`") elsewhere in the model or specified as data); and multiply defined parameters or data (for example, `y.hat<-a+b*x[i]` defined within a loop, thus implicitly creating n different definitions of the same variable `y.hat`). Many of these errors can be detected from the error messages given by Bugs.

Problems with data. Improper data in Bugs include NA's in unmodeled data (see Section 16.8); passing data that are not in the model (this can occur, for example, if a predictor `x` is removed from the model but kept in the data list); passing the wrong data (for example, if a variable name has been changed, or if an R object containing data has been overwritten); passing empty data vectors (for variables that you have forgotten to set up in R); subscripting problems (length or dimension of a data array inconsistent with its use in the Bugs model); and data out of range (for example, passing a negative value to data modeled by a lognormal distribution, or passing a fractional value to data modeled by a binomial distribution).

Problems with initial values. The list of initial values can create problems because of variables that are empty or have missing values (often because of problems in the R code); and if initial values are of range of the model (this can happen, for example, with constrained parameters, as well as more obvious scenarios such as negative variance parameters).

Problems when updating. Bugs sometimes crashes because some of its updating routines are not so robust, especially when in extreme areas of parameter space, such as continuous parameters with very large values or probabilities very close to 0 or 1. One reason we like to give moderate initial values to all the parameters in the model is to avoid these problems. When Bugs updating continues to crash,

we can sometimes fix the problem by constraining the parameters, as with the line `p.bound[i]<-max(0,min(1,p[i]))` in the logistic regression model in Section 17.4. Other times Bugs crashes or hangs because of problems in the model that were not detected in the compilation stage, for example, circular definitions (parameter `a` defined in terms of `b`, and vice versa).

Model runs, but results don't make sense. We have already discussed near the end of Section 16.9 that, when Bugs is slow to converge, it can make sense to reparameterize the model to speed the mixing of the Gibbs sampler. We discuss some such techniques later in this chapter. However, it also happens that Bugs converges but to an unreasonable answer, or to a surprising result compared to results from previously fit classical models and simple multilevel models fit using R. Nonsensical inferences commonly arise from problems in the Bugs model, such as the misspecification of a distribution (for example, confusing an inverse-variance parameter τ with a variance or standard-deviation parameter) or when parameters that should have a group-level model are assigned noninformative distributions (for example, `y[i]~dnorm(y.hat[i],tau[i])`, with a separately estimated variance parameter for each observation). Problems also arise when the data sent to the model are not what you thought they were.

Comparisons to simpler models

As illustrated in several of the examples of this book, a good way to understand and build confidence in a multilevel model is to build it up from simpler no-pooling and complete-pooling models, approximate fits using `lmer()`, and simpler versions, for example, excluding some predictors, letting some coefficients vary but not others, and so forth.

Fake-data simulation

Follow the procedure described in Section 16.7: from the estimated model, take the estimates of the unmodeled parameters as the "true values," and from these simulate "true values" of the modeled parameters and then a fake dataset. Estimate the model from the fake data, and check that the estimates for the unmodeled and modeled parameters are close to the true values.

Checking fit of model to data

An important way to build confidence in a model-fitting procedure is to check that the model fits the data. Model-checking tools include residual plots and, more generally, predictive checks where the actual data are compared, numerically or graphically, to replicated data simulated from the fitted model. We discuss this approach further in Chapter 24.

Unexpected difficulties with Bugs

Sometimes a model will fail in Bugs, even though it is only a slight modification of a successful fit. We have encountered difficulties even with pure classical regressions, when there is near-collinearity in the matrix of predictors. Ultimately we believe Bugs will be improved and these problems will no longer arise, but in the meantime we can use work-arounds such as centering predictors about their mean levels and trying different reparameterizations until something works.

19.2 General methods for reducing computational requirements

There are two ways of speeding an iterative algorithm such as the Gibbs sampler: taking less time per iteration or reducing the number of iterations to convergence. Here we briefly discuss some general approaches; then in Sections 19.3–19.5 we consider some specific methods for improving the mixing of Gibbs samplers for multilevel models.

Sampling data to speed computation

Bugs can run slowly with large datasets. Computation can be slow even for moderate-sized datasets when the number of parameters in the model is large, as can easily happen with multilevel models. For example, the polling model in Section 14.1 has more than 70 parameters (a coefficient for each of the 50 states, plus coefficients for the demographic indicators, plus hyperparameters) and about 1500 respondents. The model can be fit in a reasonable time, but when we extended it to handle data from seven different polls, totaling about 10,000 respondents, we had to wait a few minutes for each model fit. In practice, this waiting reduced the number of models we could conveniently fit, thus actually reducing the flexibility of our statistical analysis.

In this sort of setting, it can be effective to *sample* the data, for example, randomly selecting one-tenth of the respondents and analyzing them first. Once a reasonable model has been found, we can go back and take the time to fit it to the entire dataset. Sampling is easy in R; for example,

R code

```
subset <- sample (n, n/10)
n <- length(subset)
y <- y[subset]
X <- X[subset,]
state <- state[subset]
. . .
```

Computation can be sped even more using *cluster sampling*, in which we take a subset of groups, and then a sample of respondents within each group. For example, in the election poll example, we could include half the states, and one-fifth of the data within each sampled state. The advantage of cluster sampling is that it reduces the number of parameters in the model as well as the number of data points. Do not reduce the number of clusters too far, however, or it will make it difficult to estimate the group-level regression coefficients and the group-level variance. (As discussed in Section 12.9, the group-level variance is difficult to estimate if the number of groups is less than 5.)

Thinning output to save memory

Sometimes Bugs must be run a long time to convergence. This happens when the simulation draws are highly autocorrelated, so there is very little information in each draw. In this case, it makes sense to *thin* the Bugs output—that is, to just keep every k^{th} simulation draw and discard most of the rest—to save computation time and memory. We have set up the `bugs()` function to automatically thin long simulation runs so as to save approximately 1000 simulation draws.

Knowing when to give up

If you have to run Bugs for a long time and still don't reach approximate convergence, then we recommend reformulating the model using the ideas discussed in the rest of this chapter. In the meantime, you can run simpler versions of the model and see if they make sense. Our applied research often proceeds on two tracks, with a series of simple models used to explore data, while we work to get more elaborate models working reliably. Fitting an elaborate model generally becomes more possible as we get a better sense of what answers we should be expecting, based on our simpler data analysis. At the end, the more complicated model can answer questions that we did not even know to ask before we built up to it. An example is the pattern in Figure 14.11 on page 313 of state-by-state variation in the coefficient of income on vote preference, which we were only able to learn about after we succeeded in fitting the varying-intercept, varying-slope model with a group-level predictor.

19.3 Simple linear transformations

We now discuss some methods for reformulating models to make the Gibbs sampler converge in fewer iterations. As discussed in Section 18.6, the Gibbs sampler will run slowly if parameters in the model are highly correlated in their estimation—for example, if α is large, then β must be small and vice versa. This sort of correlation can easily occur between regression intercepts and slopes for predictors that are not centered at zero; see Figure 13.5 on page 288. As illustrated in that example, we improved the model by centering the predictor x before including it in the model.

Centering typically increases the speed of convergence and can be implemented easily in R without requiring any changes to the Bugs model. For example, for the simple regression

$$y_i \sim \text{N}(\alpha + \beta_1 X_{i1} + \beta_2 X_{i2},\, \sigma_y^2),$$

we can rescale the individual predictors `x1` and `x2`:

R code

```
z1 <- x1 - mean (x1)
z2 <- x2 - mean (x2)
```

and then express the Bugs model using the z's rather than x's as predictors. We then have two options: we can simply work with the fitted model in terms of the new, adjusted predictors; or we can adjust the fitted coefficients to work with the original predictors X. The fitted model is

$$\begin{aligned} y_i &= \alpha + \beta_1 Z_{i1} + \beta_2 Z_{i2} + \epsilon_i \\ &= \alpha + \beta_1 (X_{i1} - \bar{X}_1) + \beta_2 (X_{i2} - \bar{X}_2) + \epsilon_i \\ &= (\alpha - \beta_1 \bar{X}_1 - \beta_2 \bar{X}_2) + \beta_1 X_{i1} + \beta_2 X_{i2} + \epsilon_i \end{aligned}$$

Thus, the intercept on the original scale is simply $\alpha - \beta_1 \bar{X}_1 - \beta_2 \bar{X}_2$, and the slopes are unaffected.

19.4 Redundant parameters and intentionally nonidentifiable models

Another way of speeding the convergence of the Gibbs sampler involves adding redundant coefficients that are collinear with existing predictors in the model.

We have already discussed one connection between multilevel models and identifiability: in a classical regression model, we can include only $J-1$ of the J group-level indicators (with the remaining indicator serving as a baseline), whereas in multilevel

regression, we can include coefficients for all J groups, because they are modeled as coming from a common distribution with a finite variance. The group-level model provides information that allows the coefficients to be identified (as expressed mathematically by the augmented data vector y_* in (18.16) on page 397).

Our use of nonidentifiability here is different—it is purely for computational, not modeling, purposes, and is nonidentified even in the hierarchical model.

Redundant mean parameters for a simple nested model

As a simple example, we return to the radon analysis from Chapter 12, simplified to measurements within counties, ignoring all individual-level and group-level predictors. We can write the model as

$$\begin{aligned} y_i &\sim \text{N}(\mu + \eta_{j[i]}, \sigma_y^2), \text{ for } i = 1, \ldots, n \\ \eta_j &\sim \text{N}(0, \sigma_\eta^2), \text{ for } j = 1, \ldots, J. \end{aligned}$$

In Bugs:

Bugs code

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (y.hat[i], tau.y)
    y.hat[i] <- mu + eta[county[i]]
  }
  mu ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)

  for (j in 1:n.county){
    eta[j] ~ dnorm (0, tau.eta)
  }
  tau.eta <- pow(sigma.eta, -2)
  sigma.eta ~ dunif (0, 100)
}
```

With this version of the model, however, the Bugs simulations are slow to converge; see Figure 19.2. It is possible for the simulations to get stuck in a configuration where the entire vector η is far from zero (even though the η_j's are assigned a distribution with mean 0). Ultimately, the simulations will converge to the correct distribution, but we do not want to have to wait. We can speed the convergence by adding a *redundant parameter* for the mean, and then redefining new parameters η centered at an arbitrary group-level mean μ_η, thus replacing the loop `for (j in 1:n.county)` in the above model by

Bugs code

```
  mu.adj <- mu + mean(eta[])
  for (j in 1:n.county){
    eta[j] ~ dnorm (mu.eta, tau.eta)
    eta.adj[j] <- eta[j] - mean(eta[])
  }
  mu.eta ~ dnorm (0, .0001)
}
```

To fit the new model, we must specify initial values to the redundant parameters, μ_η, η_j, but we only need to save the adjusted parameters, $\mu^{\text{adj}}, \eta^{\text{adj}}$:

R code

```
radon.data <- list ("n", "y", "n.county", "county")
radon.inits <- function(){
  list (mu=rnorm(1), mu.eta=rnorm(1), eta=rnorm(n.county),
```

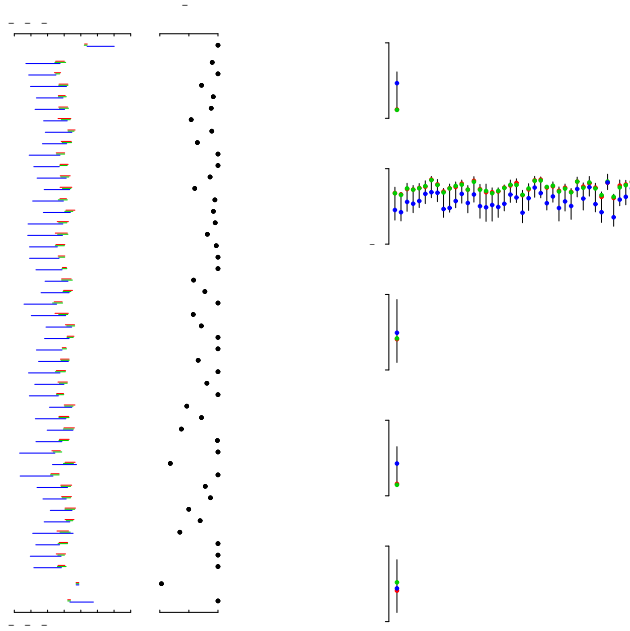

Figure 19.2 *Summary of Bugs simulations of the one-way analysis of variance for the radon data. Convergence is slow, so we used redundant parameters to help the model run more efficiently.*

```
    sigma.y=runif(1), sigma.eta=runif(1))
}
radon.parameters <- c ("mu.adj", "eta.adj", "sigma.y", "sigma.eta")
fit.4 <- bugs (radon.data, radon.inits, radon.parameters,
  "M4.bug", n.chains=3, n.iter=100)
```

Redundant mean parameters for a non-nested model: flight simulator example

We can use the same overparameterization trick for non-nested models. For example, we can express the prior distributions for the flight simulator model (13.9) as

$$\begin{aligned} \gamma_j &\sim \text{N}(\mu_\gamma, \sigma_\gamma^2), \text{ for } j = 1, \ldots, J \\ \delta_k &\sim \text{N}(\mu_\delta, \sigma_\delta^2), \text{ for } k = 1, \ldots, K. \end{aligned} \tag{19.1}$$

The means μ_γ, μ_δ, and the individual raw treatment and airport effects γ_j, δ_k are not separately identified. However, we can define the parameters of interest by centering each batch of group-level coefficients around zero and then redefining the mean. This can all be implemented in Bugs (compare to the model on page 380):

Bugs code

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

```
    y[i] ~ dnorm (y.hat[i], tau.y)
    y.hat[i] <- mu + g[treatment[i]] + d[airport[i]]
  }
  mu.adj <- mu + mean(g[]) + mean(d[])
  mu ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)

  for (j in 1:n.treatment){
    g[j] ~ dnorm (mu.g, tau.g)
    g.adj[j] <- g[j] - mean(g[])
  }
  mu.g ~ dnorm (0, .0001)
  tau.g <- pow(sigma.g, -2)
  sigma.g ~ dunif (0, 100)

  for (k in 1:n.airport){
    d[k] ~ dnorm (mu.d, tau.d)
    d.adj[k] <- d[k] - mean(d[])
  }
  mu.d ~ dnorm (0, .0001)
  tau.d <- pow(sigma.d, -2)
  sigma.d ~ dunif (0, 100)
}
```

As with the previous example, the redundant location parameters μ_γ and μ_δ can reduce the number of iterations required for the Gibbs sampler to converge for the parameters of interest: μ^{adj}, the γ_j^{adj}'s, and the δ_k^{adj}'s.

Example: multilevel logistic regression for survey responses

For a more elaborate problem in which redundant parameters are useful, we consider the example from Section 14.1 of the probability of a Yes response on a survey, estimated as a function of demographics and state of residence.

We write the model in Bugs by expanding the multilevel logistic regression model on page 381:

Bugs code

```
mu.adj <- b.0 + mean(b.age[]) + mean(b.edu[]) + mean(b.age.edu[,]) +
  mean(b.state[])
for (j in 1:n.age){
  b.age[j] ~ dnorm (mu.age, tau.age)
  b.age.adj[j] <- b.age[j] - mean(b.age[])
}
for (j in 1:n.edu){
  b.edu[j] ~ dnorm (mu.edu, tau.edu)
  b.edu.adj[j] <- b.edu[j] - mean(b.edu[])
}
for (j in 1:n.age) {for (k in 1:n.edu){
  b.age.edu[j,k] ~ dnorm (mu.age.edu, tau.age.edu)
  b.age.edu.adj[j,k] <- b.age.edu[j,k] - mean(b.age.edu[,])
}
for (j in 1:n.state){
  b.state[j] ~ dnorm (b.state.hat[j], tau.state)
  b.state.hat[j] <- b.region[region[j]] + b.v.prev*v.prev[j]
}
b.v.prev ~ dnorm (0, .0001)
```

```
for (j in 1:n.region){
  b.region[j] ~ dnorm (mu.region, tau.region)
  b.region.adj[j] <- b.region[j] - mean(b.region[])
}
```

We assign noninformative prior distributions to the new hierarchical mean parameters:

Bugs code

```
mu.age ~ dnorm (0, .0001)
mu.edu ~ dnorm (0, .0001)
mu.age.edu ~ dnorm (0, .0001)
mu.region ~ dnorm (0, .0001)
```

and retain the rest of the model from page 381. The expression for `mu.adj` does not include `mean(b.region[])` because the parameters `b.region` are not directly included in the expression for `logit(p[i])`; the region coefficients come in through the state coefficients `b.state`.

As with the other examples in this section, the redundant mean parameters, $\mu_{\text{age}}, \mu_{\text{edu}}, \mu_{\text{age.edu}}$, are not separately identified (except by their prior distributions). We will not expect them to converge rapidly in the simulations, and similarly we would expect convergence problems with the multilevel parameters β_{age}, *before* they were centered at these μ's.

A slight complexity arises in that we add a redundant mean to β_{region} but not to β_{state} to be consistent with the regional model that is nested within.

After 500 steps (in the call to `bugs()` from R, we set `n.chains=3` and `n.iter=500`), the β's and σ's have reached approximate convergence.

(A simpler way to write the model would be to replace the μ parameters in the model by zeroes, which would leave all the coefficients identifiable and eliminate the need to center the β's. However, this model can be slower to converge—setting the μ's to zero causes a kind of "gridlock" in the Gibbs sampler with multilevel coefficients.)

"Adjusted" or "raw" parameters

There are two equivalent ways of labeling in a reparameterization. The method we have described so far is to start with an existing model, add parameters to it, then define new "adjusted" parameters that are identified by the data. Thus, for example, a Bugs model would include an unidentified α and an identified α^{adj}, which is what would be saved from the fitting of the model. This approach is convenient in that it begins with an already-working model but is awkward in that the saved parameters do not have the names that were used in the statistical formulation of the model (except in cases such as (15.2) on page 326 for the study of police stops, where we explicitly define adjusted parameters of interest).

An alternative approach is to relabel the parameters in the original model as "raw" and then give the identified parameters the clean names that would be saved in the model fit. Thus, instead of `a.adj[j]<-a[j]-mean(a[])`, the Bugs model would have `a[j]<-a.raw[j]-mean(a.raw[])`. When we know ahead of time that we will be using redundant parameters or models expressed in a compound form (as in the scaled inverse-Wishart model on page 376), we often start right away with the formulation in terms of "raw" parameters to simplify the post-processing of inferences in R.

19.5 Parameter expansion: multiplicative redundant parameters

We next present a slightly more sophisticated trick involving redundant multiplicative as well as additive parameters, an idea that is a response to a particular kind of slow convergence of Gibbs samplers for hierarchical models.

The Gibbs sampler can get stuck

Gibbs samplers for multilevel models can get stuck in the following way. Suppose that a group-level variance parameter σ_α happens to be estimated near zero. Then, in the updating step for $\alpha_1, \ldots, \alpha_J$, these group-level coefficients will be pooled strongly toward their common mean (see Figure 18.4: if σ_α is near zero, then the ratio $\sigma_y^2/\sigma_\alpha^2$ will be large, and so we will be near the top of the table where there is more pooling). So the α_j's will be estimated to be close to each other. But the updating step for σ_α is based on the variance of the α_j's, and so σ_α will be estimated to be near zero, and the algorithm can get stuck. Eventually, the simulations will move through the entire distribution of σ_α, but this can require many iterations.

Solution using parameter expansion

A good way to get the Gibbs sampler unstuck in the above scenario is to be able to rescale the α_j's by multiplying the entire α vector by a constant. We can do this in a multilevel model by adding a redundant multiplicative parameter for each variance component.

For example, we can extend the flight simulator model (13.9) with two new multiplicative parameters, ξ_γ and ξ_δ and then assigning them noninformative prior distributions. The ξ parameters are not identified in the model; their role is to rescale the existing group-level parameters:

	Old	New	
Treatment effects	γ_j	$\xi_\gamma(\gamma_j - \bar{\gamma})$	for $j = 1, \ldots, J$
Airport effects	δ_k	$\xi_\delta(\delta_k - \bar{\delta})$	for $k = 1, \ldots, K$
Treatment s.d.	σ_γ	$\lvert\xi_\gamma\rvert\,\sigma_\gamma$	
Airport s.d.	σ_δ	$\lvert\xi_\delta\rvert\,\sigma_\delta$	

Implementing all this in Bugs looks elaborate but can be built gradually as an expansion of the model on page 380:

Bugs code

```
model {
  for (i in 1:n){
    y[i] ~ dnorm (y.hat[i], tau.y)
    y.hat[i] <- mu + g[treatment[i]] + d[airport[i]]
  }
  mu ~ dnorm (0, .0001)
  tau.y <- pow(sigma.y, -2)
  sigma.y ~ dunif (0, 100)

  for (j in 1:n.treatment){
    g.raw[j] ~ dnorm (mu.g.raw, tau.g.raw)
    g[j] <- xi.g*(g.raw[j] - mean(g.raw[]))
  }
  xi.g ~ dunif (0, 100)
  mu.g.raw ~ dnorm (0, .0001)
  tau.g.raw <- pow(sigma.g.raw, -2)
```

```
  sigma.g.raw ~ dunif (0, 100)
  sigma.g <- xi.g*sigma.g.raw

  for (k in 1:n.airport){
    d.raw[k] ~ dnorm (mu.d.raw, tau.d.raw)
    d[k] <- xi.d*(d.raw[k] - mean(d.raw[]))
  }
  xi.d ~ dnorm (0, .0001)
  mu.d.raw ~ dnorm (0, .0001)
  tau.d.raw <- pow(sigma.d.raw, -2)
  sigma.d.raw ~ dunif (0, 100)
  sigma.d <- abs(xi.d)*sigma.d.raw
}
```

Example: multilevel logistic regression for survey responses

We next apply multiplicative redundant parameters to a model with individual- and group-level predictors: the logistic regression for state-level opinions from national polls. In Bugs, we re-express in terms of "raw" parameters and extend the model on page 381 to include the new ξ's:

Bugs code

```
for (j in 1:n.age){
  b.age[j] <- xi.age*(b.age.raw[j] - mean(b.age.raw[]))
  b.age.raw[j] ~ dnorm (0, tau.age.raw)
}
for (j in 1:n.edu){
  b.edu[j] <- xi.edu*(b.edu.raw[j] - mean(b.edu.raw[]))
  b.edu.raw[j] ~ dnorm (0, tau.edu.raw)
}
for (j in 1:n.age){
  for (k in 1:n.edu){
    b.age.edu[j,k] <- xi.age.edu*(b.age.edu.raw[j,k] -
      mean(b.age.edu.raw[,]))
    b.age.edu.raw[j,k] ~ dnorm (0, tau.age.edu.raw)
  }
}
for (j in 1:n.state){
  b.state[j] <- xi.state*(b.state.raw[j] - mean(b.state.raw[]))
  b.state.raw[j] ~ dnorm (b.state.raw.hat[j], tau.state.raw)
  b.state.raw.hat[j] <- b.region.raw[region[j]]+b.v.prev.raw*v.prev[j]
}
b.v.prev <- xi.state*b.v.prev
b.v.prev.raw ~ dnorm (0, .0001)
for (j in 1:n.region) {
  b.region[j] <- xi.state*b.region.raw[j]
  b.region.raw[j] ~ dnorm (0, tau.region.raw)
}
tau.age.raw <- pow(sigma.age.raw, -2)
tau.edu.raw <- pow(sigma.edu.raw, -2)
tau.age.edu.raw <- pow(sigma.age.edu.raw, -2)
tau.state.raw <- pow(sigma.state.raw, -2)
tau.region.raw <- pow(sigma.region.raw, -2)

sigma.age.raw ~ dunif (0, 100)
sigma.edu.raw ~ dunif (0, 100)
sigma.age.edu.raw ~ dunif (0, 100)
```

```
sigma.state.raw ~ dunif (0, 100)
sigma.region.raw ~ dunif (0, 100)

xi.age ~ dunif (0, 100)
xi.edu ~ dunif (0, 100)
xi.age.edu ~ dunif (0, 100)
xi.state ~ dunif (0, 100)
sigma.age <- xi.age*sigma.age.raw
sigma.edu <- xi.edu*sigma.edu.raw
sigma.age.edu <- xi.age.edu*sigma.age.edu.raw
sigma.state <- xi.state*sigma.state.raw
sigma.region <- xi.state*sigma.region.raw          # not "xi.region"
```

This model is awkwardly long, but we have attempted to use parallel structure to write it cleanly.

Application to item-response and ideal-point models

We illustrate a slightly different approach to redundant parameters with the item-response or ideal-point model with a discrimination parameter (see model (14.13) on page 316 and the accompanying discussion). Here, we resolve the additive and multiplicative nonidentifiability by *shifting* the ability and difficulty parameters using the mean of the α_j's and *scaling* the ability, difficulty, and discrimination parameters using the standard deviation of the α_j's. This is equivalent to restricting the abilities to have a mean of 0 and a standard deviation of 1, but the expression using redundant parameters allows for faster convergence of the Gibbs sampler. The Bugs model can be written as

Bugs code

```
model {
  for (i in 1:n){
    y[i] ~ dbin (p.bound[i], 1)
    p.bound[i] <- max(0, min(1, p[i]))
    logit(p[i]) <- g[k[i]]*(a[j[i]] - b[k[i]])
  }
  shift <- mean(a[])
  scale <- sd(a[])

  for (j in 1:J){
    a.raw[j] ~ dnorm (mu.a.raw, tau.a.raw)
    a[j] <- (a.raw[j] - shift)/scale
  }
  mu.a.raw ~ dnorm (0, .0001)
  tau.a.raw <- pow(sigma.a.raw, -2)
  sigma.a.raw ~ dunif (0, 100)

  for (k in 1:K){
    b.raw[k] ~ dnorm (b.hat.raw[j], tau.b.raw)
    b.hat.raw[j] <- b.0.raw + d.raw*x[j]
    b[k] <- (b.raw[k] - shift)/scale
    g.raw[k] ~ dnorm (mu.g.raw, tau.g.raw)
    g[k] <- g.raw[k]*scale
  }
  b.0.raw ~ dnorm (0, .0001)
  mu.g.raw ~ dnorm (0, .0001)
  tau.b.raw <- pow(sigma.b.raw, -2)
  tau.g.raw <- pow(sigma.g.raw, -2)
```

```
    sigma.b.raw ~ dunif (0, 100)
    sigma.g.raw ~ dunif (0, 100)

    d.raw ~ dnorm (0, .0001) I(0,)
    d <- d.raw*scale
}
```

The second-to-last line in this Bugs model constrains the group-level regression coefficient δ to be positive to resolve the reflection invariance in the model, as discussed in the context of model (14.14) on page 318.

The future of redundant parameterization

Bugs (as linked from R) is currently the best general-purpose tool for multilevel modeling, but in its flexibility it cannot do everything well, and so it requires various work-arounds to run effectively in many real problems. In the not-too-distant future, we expect that Bugs or its successor programs will automatically implement reparameterizations internally, so that the user can get the benefits in efficiency without the need to set up the model expansion explicitly.

An analogy could be made to least squares computations fifty years ago, when users had to program and, if necessary, perform steps such as rescaling and pivoting to obtain numerically stable results. Now the least squares routines in linpack, R, and other packages automatically do what is necessary to compute stable least squares estimates for just about any (noncollinear) data.

19.6 Using redundant parameters to create an informative prior distribution for multilevel variance parameters

The prior distribution is sometimes said to represent your knowledge about the parameters before ("prior to") seeing the data. In practice, however, the prior distribution, along with the rest of the model, is set up after seeing data, and so we prefer to think of it as representing any information outside of the data used in the likelihood.

Here we shall consider in depth the choice of prior distribution for group-level variance parameters in multilevel models. For simplicity, we work with a basic two-level normal model of data y_{ij} with group-level coefficients α_j:

$$\begin{aligned} y_{ij} &\sim \text{N}(\mu + \alpha_j, \sigma_y^2), \quad i = 1, \ldots, n_j, \quad j = 1, \ldots, J \\ \alpha_j &\sim \text{N}(0, \sigma_\alpha^2), \quad j = 1, \ldots, J. \end{aligned} \tag{19.2}$$

We briefly discuss other hierarchical models at the end of this section.

Model (19.2) has three hyperparameters—μ, σ_y, and σ_α—but here we concern ourselves only with the last of these. Typically, enough data will be available to estimate μ and σ_y that one can use any reasonable noninformative prior distribution—for example, $p(\mu, \sigma_y) \propto 1$ or $p(\mu, \log \sigma_y) \propto 1$.

Various noninformative prior distributions have been suggested in Bayesian literature and software, including an improper uniform density on σ_α and proper distributions such as $\sigma_\alpha^2 \sim$ inverse-gamma$(0.001, 0.001)$. In this section, we explore and make recommendations for prior distributions for σ_α. We find that some purportedly noninformative prior distributions can unduly affect inferences, especially for problems where the number of groups J is small or the group-level standard deviation σ_α is close to zero.

Informative prior distributions

We can construct a rich family of prior distributions by applying a redundant multiplicative reparameterization to model (19.2):

$$\begin{aligned} y_{ij} &\sim \text{N}(\mu + \xi\eta_j, \sigma_y^2) \\ \eta_j &\sim \text{N}(0, \sigma_\eta^2). \end{aligned} \tag{19.3}$$

The parameters α_j in (19.2) correspond to the products $\xi\eta_j$ in (19.3), and the hierarchical standard deviation σ_α in (19.2) corresponds to $|\xi|\sigma_\eta$ in (19.3). The parameters ξ, η, σ_η are not separately identifiable in this model. As discussed in Section 19.5, adding the redundant multiplicative parameter ξ can speed the convergence of the Gibbs sampler.

In addition, this expanded model form allows us to construct a family of prior distributions for the hierarchical variance parameter σ_α by separately assigning prior distributions to both ξ and σ_η, thus implicitly creating a model for $\sigma_\alpha = |\xi|\sigma_\eta$.

For simplicity we restrict ourselves to independent prior distributions on ξ and σ_η, with a normal distribution for ξ and inverse-gamma for σ_η^2. (These are technically known as *conditionally conjugate* prior distributions, for reasons that are not relevant to us here.)

The implicit conditionally conjugate family for σ_α is then the set of distributions corresponding to the absolute value of a normal random variable, divided by the square root of a gamma random variable. That is, σ_α has the distribution of the absolute value of a noncentral-t variate. We call this the *folded noncentral t distribution*, with the "folding" corresponding to the absolute value operator. The noncentral t in this context has three parameters, which can be identified with the mean of the normal distribution for ξ, and the scale and degrees of freedom for σ_η^2. (Without loss of generality, the scale of the normal distribution for ξ can be set to 1 since it cannot be separated from the scale for σ_η.)

The folded noncentral t distribution is not commonly used in statistics, and we find it convenient to understand it through various special and limiting cases. In the limit that the denominator is specified exactly, we have a folded normal distribution; conversely, specifying the numerator exactly yields the square-root-inverse-χ^2 distribution for σ_α.

An appealing two-parameter family of prior distributions is determined by restricting the prior mean of the numerator to zero, so that the folded noncentral t distribution for σ_α becomes simply a half-t—that is, the absolute value of a Student-t distribution centered at zero. We can parameterize this in terms of scale s_α and degrees of freedom ν:

$$p(\sigma_\alpha) \propto \left(1 + \frac{1}{\nu}\left(\frac{\sigma_\alpha}{s_\alpha}\right)^2\right)^{-(\nu+1)/2}$$

This family includes, as special cases, the improper uniform density (if $\nu = -1$) and the proper half-Cauchy, $p(\sigma_\alpha) \propto \left(\sigma_\alpha^2 + s_\alpha^2\right)^{-1}$ (if $\nu = 1$).

Noninformative prior distributions

Uniform prior distributions. We first consider uniform prior distributions while recognizing that we must be explicit about the scale on which the distribution is defined. Various choices have been proposed for modeling variance parameters. A uniform prior distribution on $\log \sigma_\alpha$ would seem natural—working with the logarithm of a parameter that must be positive—but it results in an improper posterior

distribution. An alternative would be to define the prior distribution on a compact set (for example, in the range $[-A, A]$ for some large value of A), but then the posterior distribution would depend strongly on $-A$, the lower bound of the prior support.

The problem arises because the marginal likelihood, $p(y|\sigma_\alpha)$—after integrating over α, μ, σ_y in (19.2)—approaches a finite nonzero value as $\sigma_\alpha \to 0$. Thus, if the prior density for $\log \sigma_\alpha$ is uniform, the posterior distribution will have infinite mass integrating to the limit $\log \sigma_\alpha \to -\infty$. To put it another way, in a hierarchical model the data can never rule out a group-level variance of zero, and so the prior distribution should not put an infinite mass in this area.

Another option is a uniform prior distribution on σ_α itself, which has a finite integral near $\sigma_\alpha = 0$ and thus avoids the above problem. We have generally used this noninformative density in our applied work, but it has a slightly disagreeable miscalibration toward positive values (see footnote 1 on page 433), with its infinite prior mass in the range $\sigma_\alpha \to \infty$. With $J = 1$ or 2 groups, this actually results in an improper posterior density, essentially concluding $\sigma_\alpha = \infty$ and doing no pooling. In a sense this is reasonable behavior, since it would seem difficult from the data alone to decide how much, if any, pooling should be done with data from only one or two groups. However, from a Bayesian perspective it is awkward for the decision to be made ahead of time, as it were, with the data having no say in the matter. In addition, for small J, such as 4 or 5, we worry that the heavy right tail of the posterior distribution would lead to overestimates of σ_α and thus result in pooling that is less than optimal for estimating the individual α_j's.

We can interpret the various improper uniform prior densities as limits of proper distributions. The uniform density on $\log \sigma_\alpha$ is equivalent to $p(\sigma_\alpha) \propto \sigma_\alpha^{-1}$ or $p(\sigma_\alpha^2) \propto \sigma_\alpha^{-2}$, which has the form of an inverse-χ^2 density with 0 degrees of freedom and can be taken as a limit of proper inverse-gamma densities.

The uniform density on σ_α is equivalent to $p(\sigma_\alpha^2) \propto \sigma_\alpha^{-1}$, an inverse-$\chi^2$ density with -1 degrees of freedom. This density cannot easily be seen as a limit of proper inverse-χ^2 densities (since these must have positive degrees of freedom), but it can be interpreted as a limit of the half-t family on σ_α, where the scale approaches ∞ (and any value of ν). Or, in the expanded notation of (19.3), one could assign any prior distribution to σ_η and a normal to ξ, and let the prior variance for ξ approach ∞.

Another noninformative prior distribution sometimes proposed in the Bayesian literature is uniform on σ_α^2. We do not recommend this, as it seems to have the miscalibration toward higher values as described above, but more so, and also requires $J \geq 4$ groups for a proper posterior distribution.

Inverse-gamma(ϵ, ϵ) prior distributions. The inverse-gamma(ϵ, ϵ) prior distribution is an attempt at noninformativeness within the conditionally conjugate family, with ϵ set to a low value such as 1 or 0.01 or 0.001 (the latter value being used in some of the examples in Bugs). A difficulty of this prior distribution is that in the limit of $\epsilon \to 0$ it yields an improper posterior density, and thus ϵ must be set to a reasonable value. Unfortunately, for datasets in which low values of σ_α are possible, inferences become very sensitive to ϵ in this model, and the prior distribution hardly looks noninformative, as we illustrate next. So we do not recommend the use of this inverse-gamma model as a noninformative prior distribution.

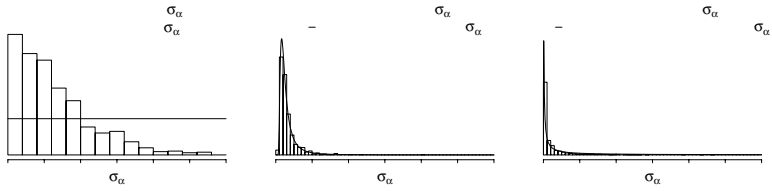

Figure 19.3 *Histograms of posterior simulations of the between-school standard deviation, σ_α, from models with three different prior distributions: (a) uniform prior distribution on σ_α, (b) inverse-gamma$(1, 1)$ prior distribution on σ_α^2, (c) inverse-gamma$(0.001, 0.001)$ prior distribution on σ_α^2. Overlain on each is the corresponding prior density function for σ_α. (For models (b) and (c), the density for σ_α is calculated using the gamma density function multiplied by the Jacobian (see the footnote on page 409) of the $1/\sigma_\alpha^2$ transformation.) We prefer the uniform prior distribution shown in the left plot. In the center and right plots, posterior inferences are strongly constrained by the prior distribution. Adapted from Gelman et al. (2003, appendix C).*

Example: educational testing experiments in 8 schools

We demonstrate the properties of some proposed noninformative prior densities with a simple example of data from $J = 8$ educational testing experiments. Here, the parameters $\alpha_1, \ldots, \alpha_8$ represent the relative effects of Scholastic Aptitude Test coaching programs in 8 different schools, and σ_α represents the between-school standard deviations of these effects. The effects are measured as points on the test, which was scored from 200 to 800; thus the largest possible range of effects could be 600 points, with a realistic upper limit on σ_α of 100, say.

Here is the Bugs code for the model with half-Cauchy prior distribution:

Bugs code

```
model {
  for (j in 1:J){                          # J = the number of schools
    y[j] ~ dnorm (theta[j], tau.y[j])      # data model:  the likelihood
    theta[j] <- mu.theta + xi*eta[j]
    tau.y[j] <- pow(sigma.y[j], -2)
  }
  xi ~ dnorm (0, tau.xi)
  tau.xi <- pow(prior.scale, -2)
  for (j in 1:J){
    eta[j] ~ dnorm (0, tau.eta)            # hierarchical model for theta
  }
  tau.eta ~ dgamma (.5, .5)                # chi^2 with 1 d.f.
  sigma.theta <- abs(xi)/sqrt(tau.eta)     # cauchy = normal/sqrt(chi^2)
  mu.theta ~ dnorm (0, .0001)              # noninformative prior on mu
}
```

When running this model from R, we set `prior.scale <- 25`; give `y, sigma.y, J, prior.scale` as data for the `bugs()` call; and initialize the parameters `eta, xi, mu.theta, tau.eta`.

Figure 19.3 shows the posterior distributions for the 8-schools model resulting from three different choices of prior distributions that are intended to be noninformative.

The leftmost histogram of Figure 19.3 displays the posterior distribution for σ_α (as represented by 6000 simulation draws from a model fit using Bugs) for the model with uniform prior density. The data show support for a range of values

below $\sigma_\alpha = 20$, with a slight tail after that, reflecting the possibility of larger values, which are difficult to rule out given that the number of groups J is only 8—that is, not much more than the $J = 3$ required to ensure a proper posterior density with finite mass in the right tail.

In contrast, the middle histogram in Figure 19.3 shows the result with an inverse-gamma$(1, 1)$ prior distribution for σ_α^2. This new prior distribution leads to changed inferences. In particular, the posterior mean and median of σ_α are lower and pooling of the α_j's is greater than in the previously fitted model with a uniform prior distribution on σ_α. To understand this, it helps to graph the prior distribution in the range for which the posterior distribution is substantial. The graph shows that the prior distribution is concentrated in the range $[0.5, 5]$, a narrow zone in which the likelihood is close to flat compared to this prior (as we can see because the distribution of the posterior simulations of σ_α closely matches the prior distribution, $p(\sigma_\alpha)$). By comparison, in the left graph, the uniform prior distribution on σ_α seems closer to "noninformative" for this problem, in the sense that it does not appear to be constraining the posterior inference.

Finally, the rightmost histogram in Figure 19.3 shows the corresponding result with an inverse-gamma$(0.001, 0.001)$ prior distribution for σ_α^2. This prior distribution is even more sharply peaked near zero and further distorts posterior inferences, with the problem arising because the marginal likelihood for σ_α remains high near zero.

In this example, we do not consider a uniform prior density on $\log \sigma_\alpha$, which would yield an improper posterior density with a spike at $\sigma_\alpha = 0$, like the rightmost graph in Figure 19.3, but more so. We also do not consider a uniform prior density on σ_α^2, which would yield a posterior distribution similar to the leftmost graph in Figure 19.3, but with a slightly higher right tail.

This example is a gratifying case in which the simplest approach—the uniform prior density on σ_α—seems to perform well. This model is also straightforward to program directly using the Gibbs sampler or in Bugs, using either the basic model (19.2) or using the expanded parameterization (19.3), which converges slightly faster.

The appearance of the histograms and density plots in Figure 19.3 is crucially affected by the choice to plot them on the scale of σ_α. If instead they were plotted on the scale of $\log \sigma_\alpha$, the inverse-gamma$(0.001, 0.001)$ prior density would appear to be the flattest. However, the inverse-gamma(ϵ, ϵ) prior is not at all "noninformative" for this problem since the resulting posterior distribution remains highly sensitive to the choice of ϵ. The hierarchical model likelihood does not constrain $\log \sigma_\alpha$ in the limit $\log \sigma_\alpha \to -\infty$, and so a prior distribution that is noninformative on the log scale will not work.

Weakly informative prior distribution for the 3-schools problem

The uniform prior distribution seems fine for the 8-schools analysis, but problems arise if the number of groups J is much smaller, in which case the data supply little information about the group-level variance, and a noninformative prior distribution can lead to a posterior distribution that is improper or is proper but unrealistically broad.

We demonstrate by reanalyzing the 8-schools example using the data from just 3 of the schools.

Figure 19.4 displays the inferences for σ_α from two different prior distributions. First we continue with the default uniform distribution that worked well with $J = 8$

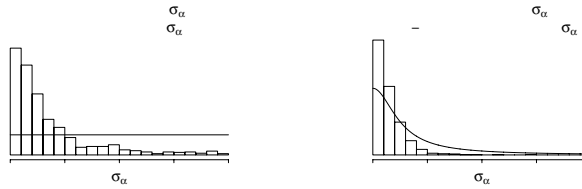

Figure 19.4 *Histograms of posterior simulations of the between-school standard deviation, σ_α, from models for the 3-schools data with two different prior distributions on σ_α: (a) uniform $(0, \infty)$, (b) half-Cauchy with scale 25, set as a weakly informative prior distribution given that σ_α was expected to be well below 100. The histograms are not on the same scales. Overlain on each histogram is the corresponding prior density function. With only $J = 3$ groups, the noninformative uniform prior distribution is too weak, and the proper Cauchy distribution works better, without appearing to distort inferences in the area of high likelihood.*

(as seen in Figure 19.3). Unfortunately, as the left histogram of Figure 19.4 shows, the resulting posterior distribution for the 3-schools dataset has an extremely long right tail, containing values of σ_α that are too high to be reasonable. This heavy tail is expected since J is so low (if J were any lower, the right tail would have an infinite integral). Using this posterior distribution for σ_α will have the effect of undershrinking the estimates of the school effects α_j. Better estimates should be obtained by including some information about σ_α that will restrict it to a more realistic range.

The right histogram of Figure 19.4 shows the posterior inference for σ_α resulting from a half-Cauchy prior distribution with scale parameter 25. As the line on the graph shows, this prior distribution is close to flat over the plausible range of $\sigma_\alpha < 50$, falling off gradually beyond this point. We call this prior distribution "weakly informative" on this scale because, even at its tail, it has a gentle slope (unlike, for example, a half-normal distribution) and can let the data dominate if the likelihood is strong in that region. This prior distribution performs well in this example, reflecting the marginal likelihood for σ_α at its low end but removing much of the unrealistic upper tail.

This half-Cauchy prior distribution would also perform well in the 8-schools problem, but it was unnecessary because the default uniform prior gave reasonable results. With only 3 schools, we went to the trouble of using a weakly informative prior, a distribution that was not intended to represent our actual prior state of knowledge about σ_α but rather to constrain the posterior distribution, to an extent allowed by the data.

General comments

Prior distributions for variance parameters. In fitting hierarchical models, we recommend starting with a noninformative uniform prior density on standard-deviation parameters σ_α. We expect this will generally work well unless the number of groups J is low (below 5, say). If J is low, the uniform prior density tends to lead to high estimates of σ_α, as discussed above. This miscalibration is an unavoidable consequence of the asymmetry in the parameter space, with variance parameters re-

stricted to be positive. Similarly, there are no always-nonnegative classical unbiased estimators of σ_α or σ_α^2 in the hierarchical model.[1]

For a noninformative but proper prior distribution, we recommend approximating the uniform density on σ_α by a uniform on a wide range (for example, uniform from 0 to 100 in the SAT coaching example) or a half-normal centered at 0 with standard deviation set to a high value such as 100. The latter approach is particularly easy to program as a $\mathrm{N}(0, 100^2)$ prior distribution for ξ in (19.3).

When more prior information is desired, for instance to restrict σ_α away from very large values, we recommend working within the half-t family of prior distributions, which are more flexible and have better behavior near 0, compared to the inverse-gamma family. A reasonable starting point is the half-Cauchy family, with scale set to a value that is high but not off the scale, for example, 25 in the SAT coaching example.

We do *not* recommend the inverse-gamma(ϵ, ϵ) family of noninformative prior distributions because, as discussed in Sections 3.3 and 4.1, in cases where σ_α is estimated to be near zero, the resulting inferences will be sensitive to ϵ. The setting of near-zero variance parameters is important partly because this is where uncertainty in the variance parameters is particularly relevant for multilevel inference.

Figure 19.3 illustrates the generally robust properties of the uniform prior density on σ_α. Many Bayesians have preferred the inverse-gamma prior family, possibly because its conditional conjugacy suggested clean mathematical properties. However, by writing the hierarchical model in the form (19.3), we see conditional conjugacy in the wider class of half-t distributions on σ_α, which include the uniform and half-Cauchy densities on σ_α (as well as inverse-gamma on σ_α^2) as special cases. From this perspective, the inverse-gamma family has nothing special to offer, and we prefer to work on the scale of the standard deviation parameter σ_α, which is typically directly interpretable in the original model.

Application to other models. The reasoning in this paper should apply to multilevel models in general. The key idea is that parameters α_j—in general, group-level exchangeable parameters—have a common distribution with some scale parameter, which we label σ_α. In addition, when group-level regression predictors must be estimated, more than $J = 3$ groups may be necessary to estimate σ_α from a noninformative prior distribution, thus requiring at least weakly informative prior distributions for the regression coefficients, the variance parameters, or both.

[1] More formally, we can evaluate the inferences using the concept of *calibration* of the posterior mean, the Bayesian analogue to the classical notion of "bias." For any parameter θ, we label the posterior mean as $\hat{\theta} = \mathrm{E}(\theta|y)$ and define the *miscalibration* of the posterior mean as $\mathrm{E}(\theta|\hat{\theta}, y) - \hat{\theta}$, for any value of $\hat{\theta}$. If the prior distribution is true—that is, if the data are constructed by first drawing θ from $p(\theta)$, then drawing y from $p(y|\theta)$—then the posterior mean is automatically calibrated; that is its miscalibration is 0 for all values of $\hat{\theta}$.
For improper prior distributions, however, things are not so simple, since it is impossible for θ to be drawn from an unnormalized density. To evaluate calibration in this context, it is necessary to posit a "true prior distribution" from which θ is drawn along with the "inferential prior distribution" that is used in the Bayesian inference.
For the hierarchical model discussed here, we can consider the improper uniform density on σ_α as a limit of uniform prior densities on the range $(0, A)$, with $A \to \infty$. For any finite value of A, we can then see that the improper uniform density leads to inferences with a positive miscalibration—that is, overestimates (on average) of σ_α.
We demonstrate this miscalibration in two steps. First, suppose that both the true and inferential prior distributions for σ_α are uniform on $(0, A)$. Then the miscalibration is trivially zero. Now keep the true prior distribution at $\mathrm{U}(0, A)$ and let the inferential prior distribution go to $\mathrm{U}(0, \infty)$. This will necessarily increase $\hat{\theta}$ for any data y (since we are now averaging over values of θ in the range $[A, \infty)$) without changing the true θ, thus causing the average value of the miscalibration to become positive. Similar issues are discussed by Meng and Zaslavsky (2002).

Further work needs to be done in developing the next level of hierarchical models, in which there are several batches of exchangeable parameters, each with their own variance parameter; we present a simple sort of hierarchical model in Section 22.6 in the context of the analysis of variance.

19.7 Bibliographic note

Gelfand, Sahu, and Carlin (1995), Boscardin (1996), Roberts and Sahu (1997), Gelfand and Sahu (1999), and Sargent, Hodges, Carlin (2000) discuss additive transformations (also called *hierarchical centering*) and related ideas for speeding Gibbs sampler convergence. The properties of redundant multiplicative parameters have been studied by Liu, Rubin, and Wu (1998), Liu and Wu (1999), van Dyk and Meng (2001), and Gelman, Huang, et al. (2006). Gelman et al. (2003, appendix C) present R and Bugs implementations of redundant multiplicative parameterization for a simple multilevel model. Geweke (2004) and Cook, Gelman, and Rubin (2006) present some methods for using fake-data simulation to validate Bayesian software.

The treatment in Section 19.6 of prior distributions for multilevel variance parameters comes from Gelman (2006). The 8-schools example is discussed in detail in Gelman et al. (2003, chapter 5).

19.8 Exercises

1. Fake-data simulation: take an example of a multilevel model from an exercise in one of the previous chapters. Check the fitting using a fake-data simulation:
   (a) Specify the unmodeled parameters (that is, those with noninformative prior distributions in the Bugs model; see Figure 16.4 on page 366), then simulate the modeled parameters, then simulate a fake dataset.
   (b) Fit the model to the fake data and check numerically that the inferences for the unmodeled parameters are consistent with their "true values" you chose in part (a). Check graphically that the inferences for the modeled parameters are consistent with their "true values" you simulated in part (a).
2. Redundant parameterization: take a varying-intercept model from an exercise in one of the previous chapters.
   (a) Fit the model without redundant parameterization.
   (b) Include redundant additive parameters.
   (c) Include redundant additive and multiplicative parameters.
   (d) Check that the different forms of the model give the same inferences. Compare how long (in actual time, not just number of iterations) it takes for each of the simulations to reach approximate convergence.
3. Prior distributions for multilevel variance parameters: consider the varying-intercept radon model with floor of measurement as an individual-level predictor and log uranium as a county-level predictor. Data are in the folder `radon`.
   (a) Fit the model in Bugs using the different prior distributions for the group-level variance parameter discussed in Section 19.6. Compare the inferences for the group-level standard deviation and for a selection of the intercept parameters.
   (b) Repeat (a) but just analyzing the subset of the data corresponding to the first eight counties.
   (c) Repeat just using the data from the first three counties.