

## CHAPTER 17

# Metric Predicted Variable with One Metric Predictor

### Contents

17.1. Simple Linear Regression . . . . .	478
17.2. Robust Linear Regression . . . . .	479
17.2.1 Robust linear regression in JAGS . . . . .	483
17.2.1.1 Standardizing the data for MCMC sampling . . . . .	484
17.2.2 Robust linear regression in Stan . . . . .	487
17.2.2.1 Constants for vague priors . . . . .	487
17.2.3 Stan or JAGS? . . . . .	488
17.2.4 Interpreting the posterior distribution . . . . .	489
17.3. Hierarchical Regression on Individuals Within Groups . . . . .	490
17.3.1 The model and implementation in JAGS . . . . .	491
17.3.2 The posterior distribution: Shrinkage and prediction . . . . .	495
17.4. Quadratic Trend and Weighted Data . . . . .	495
17.4.1 Results and interpretation . . . . .	499
17.4.2 Further extensions . . . . .	500
17.5. Procedure and Perils for Expanding a Model . . . . .	501
17.5.1 Posterior predictive check . . . . .	501
17.5.2 Steps to extend a JAGS or Stan model . . . . .	502
17.5.3 Perils of adding parameters . . . . .	503
17.6. Exercises . . . . .	504

*The agri-bank's threatnin' to revoke my lease  
If my field's production don't rapid increase.  
Oh Lord how I wish I could divine the trend,  
Will my furrows deepen? and Will my line end?*<sup>1</sup>

In this chapter, we consider situations such as predicting a person's weight from their height, or predicting their blood pressure from their weight, or predicting their income from years of education. In these situations, the predicted variable is metric, and the single predictor is also metric.

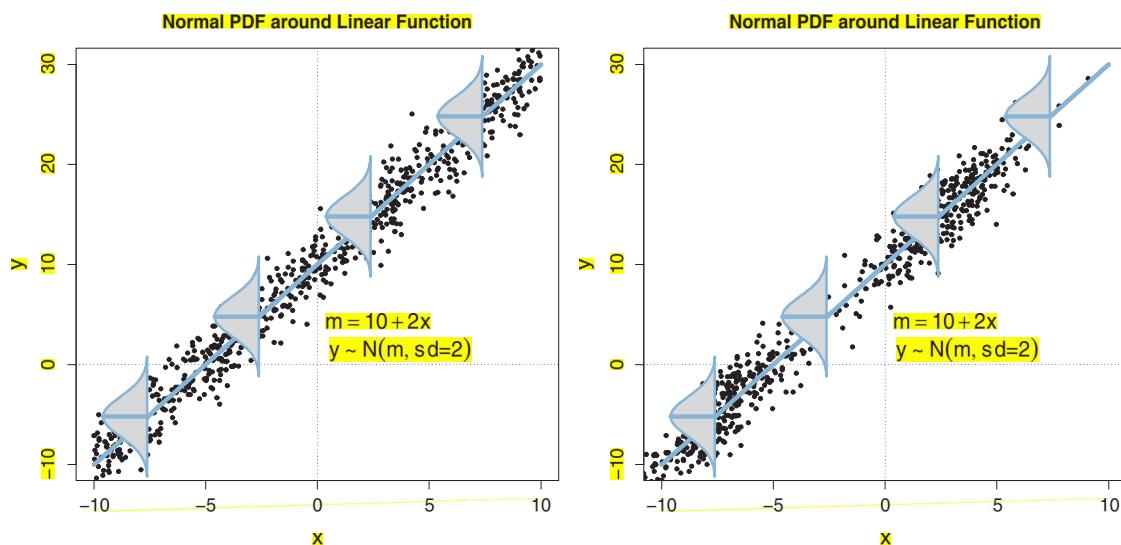
<sup>1</sup> This chapter discusses linear regression (among other topics). Analysts who use linear regression are sometimes interested in extrapolating a trend line into the future. In the poem, deeper furrows in the field might mean fresh tilling, while deeper furrows on the forehead would mean worsening worries. The “end of the line” means one thing for simple linear regression, but it can also refer to family lineage.

We will initially describe the relationship between the predicted variable,  $y$ , and predictor,  $x$ , with a simple linear model and normally distributed residual randomness in  $y$ . This model is often referred to as “simple linear regression.” We will generalize the model in three ways. First, we will give it a noise distribution that accommodates outliers, which is to say that we will replace the normal distribution with a  $t$  distribution as we did in the previous chapter. The model will be implemented in both JAGS and Stan. Next, we will consider differently shaped relations between the predictor and the predicted, such as quadratic trend. Finally, we will consider hierarchical models of situations in which every individual has data that can be described by an individual trend, and we also want to estimate group-level typical trends across individuals.

In the context of the generalized linear model (GLM) introduced in Chapter 15, this chapter’s situation involves a linear function of a single metric predictor, as indicated in the third column of Table 15.1 (p. 434), with a link function that is the identity along with a normal distribution for describing noise in the data, as indicated in the first row of Table 15.2 (p. 443). For a reminder of how this chapter’s combination of predicted and predictor variables relates to other combinations, see Table 15.3 (p. 444).

## 17.1. SIMPLE LINEAR REGRESSION

Figure 17.1 shows examples of simulated data generated from the model for simple linear regression. First a value of  $x$  is arbitrarily generated. At that value of  $x$ , the mean



**Figure 17.1** Examples of points normally distributed around a linear function. (The left panel repeats Figure 15.9, p. 442.) The model assumes that the data  $y$  are normally distributed vertically around the line, as shown. Moreover, the variance of  $y$  is the same at all values of  $x$ . The model puts no constraints on the distribution of  $x$ . The right panel shows a case in which  $x$  are distributed bimodally, whereas in the left panel the  $x$  are distributed uniformly. In both panels, there is homogeneity of variance.

predicted value of  $y$  is computed as  $\mu = \beta_0 + \beta_1 x$ . Then, a random value for the datum  $y$  is generated from a normal distribution centered at  $\mu$  with standard deviation  $\sigma$ . For a review of how to interpret the slope,  $\beta_1$ , and intercept,  $\beta_0$ , see Figure 15.1 (p. 425).

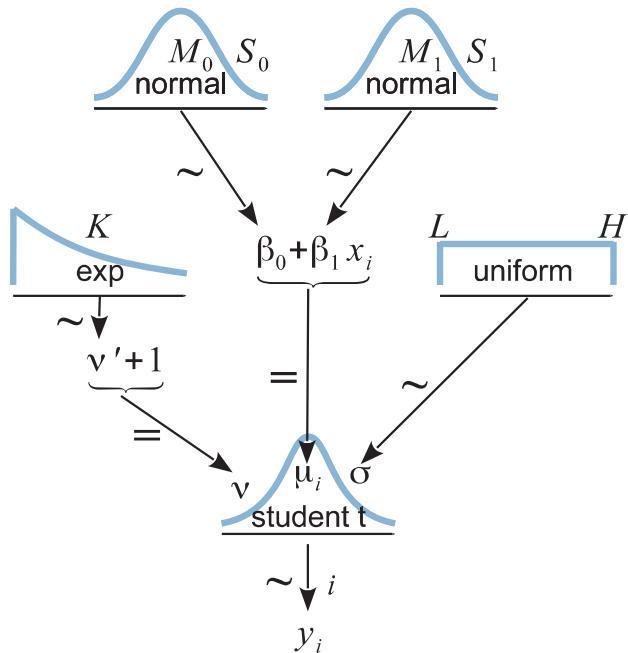
Note that the model only specifies the dependency of  $y$  on  $x$ . The model does not say anything about what generates  $x$ , and there is no probability distribution assumed for describing  $x$ . The  $x$  values in the left panel of Figure 17.1 were sampled randomly from a uniform distribution, merely for purposes of illustration, whereas the  $x$  values in the right panel of Figure 17.1 were sampled randomly from a bimodal distribution. Both panels show data from the same model of the dependency of  $y$  on  $x$ .

It is important to emphasize that the model assumes *homogeneity of variance*: At every value of  $x$ , the variance of  $y$  is the same. This homogeneity of variance is easy to see in the left panel of Figure 17.1: The smattering of data points in the vertical,  $y$ , direction appears visually to be the same at all values of  $x$ . But that is only because the  $x$  values are uniformly distributed. Homogeneity of variance is less easy to identify visually when the  $x$  values are not uniformly distributed. For example, the right panel of Figure 17.1 displays data that may appear to violate homogeneity of variance, because the apparent vertical spread of the data seems to be larger at  $x = 2.5$  than at  $x = 7.5$  (for example). Despite this deceiving appearance, the data do respect homogeneity of variance. The reason for the apparent violation is that for regions in which  $x$  is sparse, there is less opportunity for the sampled  $y$  values to come from the tails of the noise distribution. In regions where  $x$  is dense, there are many opportunities for  $y$  to come from the tails.

In applications, the  $x$  and  $y$  values are provided by some real-world process. In the real-world process, there might or might not be any direct causal connection between  $x$  and  $y$ . It might be that  $x$  causes  $y$ , or  $y$  causes  $x$ , or some third factor causes both  $x$  and  $y$ , or  $x$  and  $y$  have no causal connection, or some combination of those from multiple types of causes. The simple linear model makes no claims about causal connections between  $x$  and  $y$ . The simple linear model merely describes a tendency for  $y$  values to be linearly related to  $x$  values, hence “predictable” from the  $x$  values. When describing data with this model, we are starting with a scatter plot of points generated by an unknown process in the real world, and estimating parameter values that would produce a smattering of points that might mimic the real data. Even if the descriptive model mimics the data well (and it might not), the mathematical “process” in the model may have little if anything to do with the real-world process that created the data. Nevertheless, the parameters in the descriptive model are meaningful because they describe tendencies in the data.

## 17.2. ROBUST LINEAR REGRESSION

There is no requirement to use a normal distribution for the noise distribution. The normal distribution is traditional because of its relative simplicity in mathematical derivations. But real data may have outliers, and the use of (optionally) heavy-tailed noise



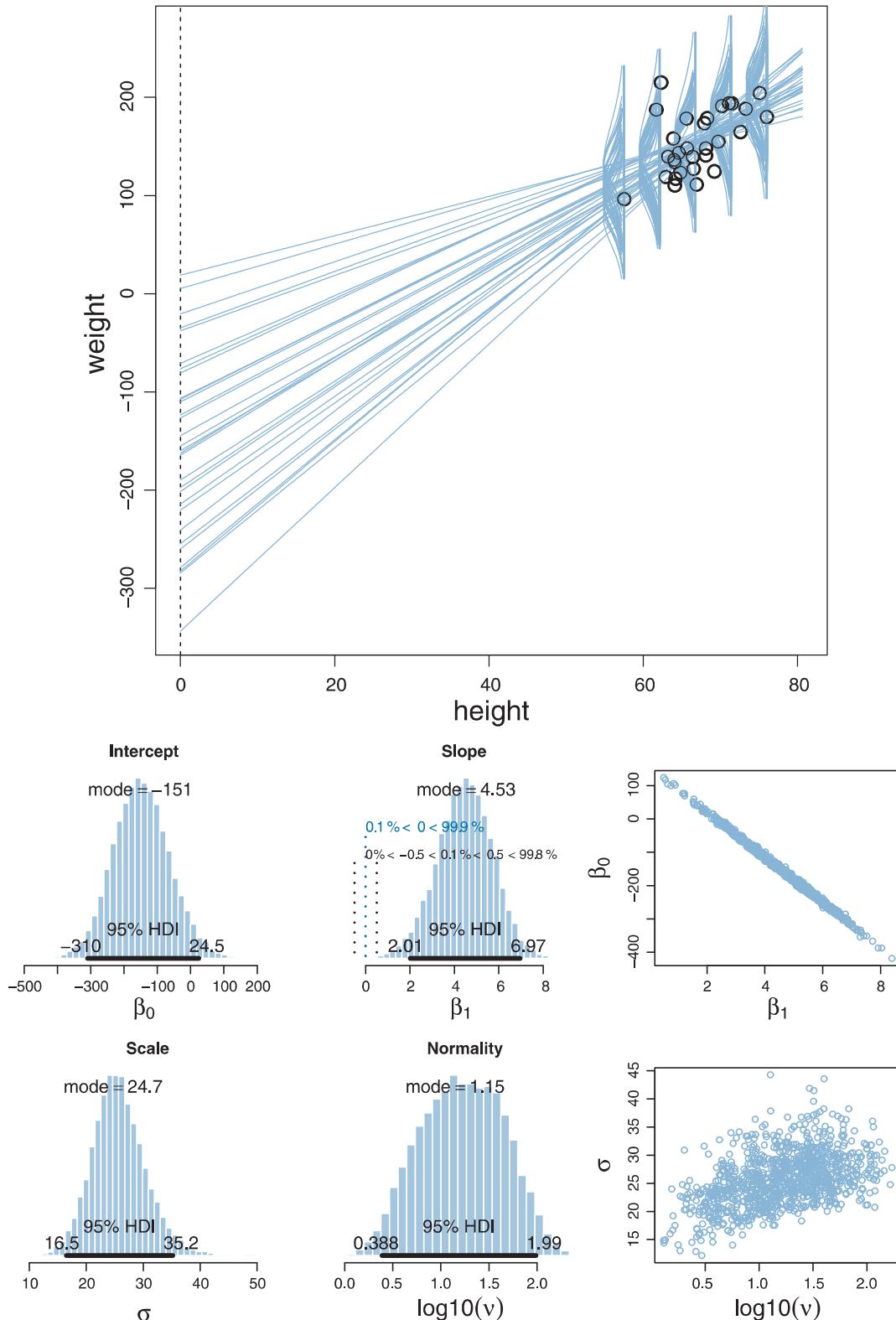
**Figure 17.2** A model of dependencies for robust linear regression. The datum,  $y_i$  at the bottom of the diagram, is distributed around the central tendency  $\mu_i$ , which is a linear function of  $x_i$ . Compare with Figure 16.11 on p. 468.

distributions is straight forward in contemporary Bayesian software. Section 16.2 (p. 458) explained the  $t$  distribution and its usefulness for describing data that might have outliers.

Figure 17.2 illustrates the hierarchical dependencies in the model. At the bottom of the diagram, the datum  $y_i$  is a  $t$ -distributed random value around the central tendency  $\mu_i = \beta_0 + \beta_1 x_i$ . The rest of the diagram illustrates the prior distribution on the four parameters. The scale parameter  $\sigma$  is given a noncommittal uniform prior, and the normality parameter  $\nu$  is given a broad exponential prior, as they were in the previous chapter. The intercept and slope are given broad normal priors that are vague on the scale of the data. Figure 17.2 is just like the diagram for robust estimation of the central tendency of groups, shown in Figure 16.11 on p. 468, except here the two parameters  $\beta_0$  and  $\beta_1$  are shown with separate normal priors, whereas in Figure 16.11, the two parameters  $\mu_1$  and  $\mu_2$  had their normal priors superimposed.

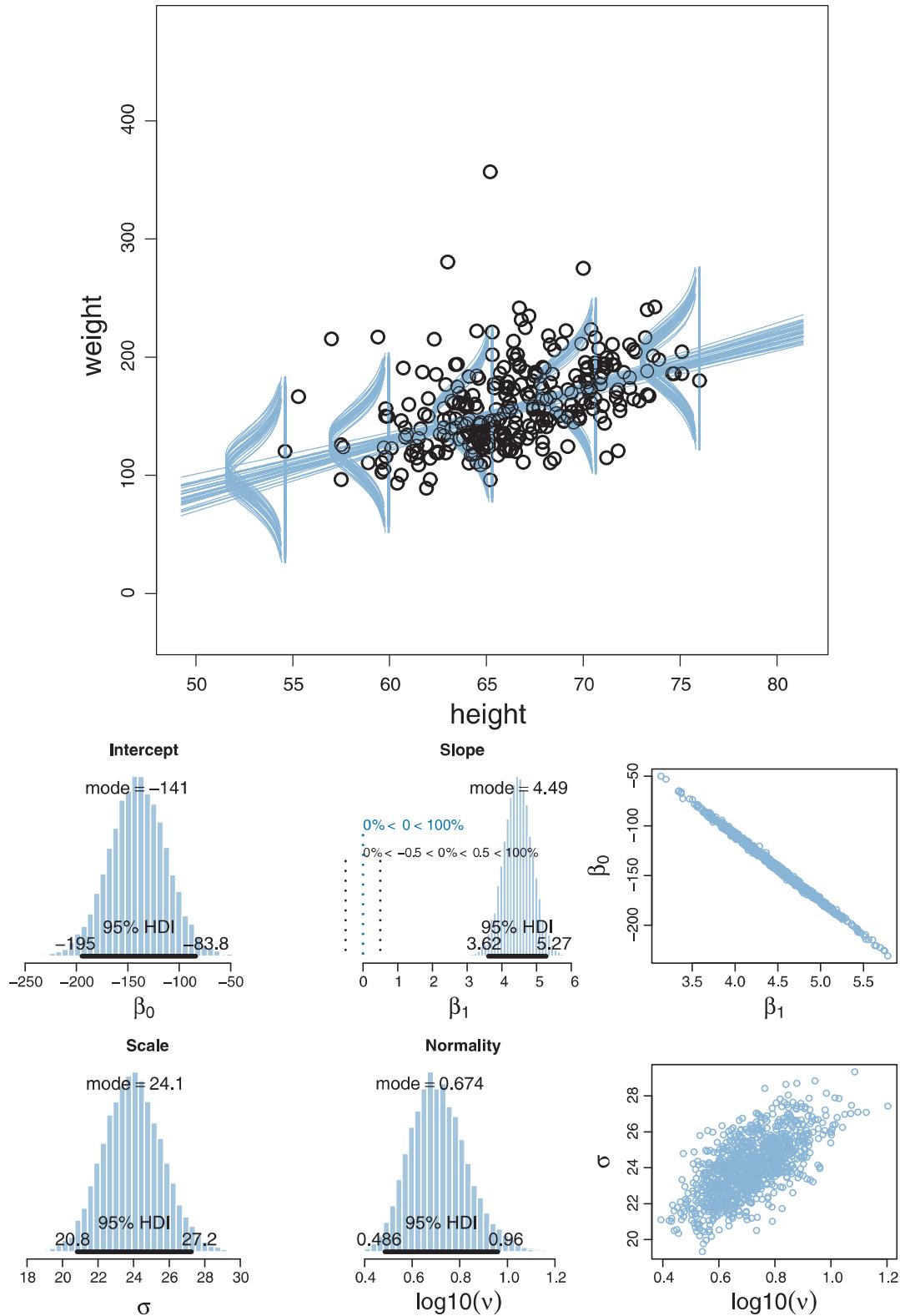
As an example, suppose we have measurements of height, in inches, and weight, in pounds, for some randomly selected people. Figures 17.3 and 17.4 show two examples, for data sets of  $N = 30$  and  $N = 300$ . At this time, focus your attention only on the scatter of data points in the top panels of each figure. The superimposed lines and curves will be explained later. You can see that the axes of the top panels are labeled height and weight, so each point represents a person with a particular combination of height and weight. The data were generated from a program I created (HtWtDataGenerator.R) that uses realistic population parameters (Brainard & Burmester, 1992). The simulated data

Data w. Post. Pred. & 95% HDI



**Figure 17.3** Upper panel: Data ( $N = 30$ ) with a smattering of credible regression lines and  $t$ -noise distributions superimposed. Lower panels: Marginal posterior distribution on parameters. Compare with [Figure 17.4](#).

Data w. Post. Pred. & 95% HDI



**Figure 17.4** Upper panel: Data ( $N = 300$ ) with a smattering of credible regression lines and t-noise distributions superimposed. Lower panels: Marginal posterior distribution on parameters. Compare with [Figure 17.3](#).

are actually based on three bivariate Gaussian clusters, not a single linear dependency. But we can still attempt to describe the data with linear regression. The scatter plots in Figures 17.3 and 17.4 do suggest that as height increases, weight also tends to increase.

Our goal is to determine what combinations of  $\beta_0$ ,  $\beta_1$ ,  $\sigma$ , and  $\nu$  are credible, given the data. The answer comes from Bayes' rule:

$$p(\beta_0, \beta_1, \sigma, \nu | D) = \frac{p(D|\beta_0, \beta_1, \sigma, \nu) p(\beta_0, \beta_1, \sigma, \nu)}{\int \int \int d\beta_0 d\beta_1 d\sigma d\nu p(D|\beta_0, \beta_1, \sigma, \nu) p(\beta_0, \beta_1, \sigma, \nu)}$$

Fortunately, we do not have to worry much about analytical derivations because we can let JAGS or Stan generate a high resolution picture of the posterior distribution. Our job, therefore, is to specify sensible priors and to make sure that the MCMC process generates a trustworthy posterior sample that is converged and well mixed.

### 17.2.1. Robust linear regression in JAGS

As has been emphasized (e.g., at the end of Section 8.2), hierarchical dependency diagrams like Figure 17.2 are useful not only for conceptualizing the structure of a model but also for programming it. Every arrow in Figure 17.2 has a corresponding line of code in JAGS or Stan. The following sections describe implementations of the model. The core of the model specification for JAGS is virtually an arrow-to-line transliteration of the diagram. The seven arrows in Figure 17.2 have seven corresponding lines of code:

```
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dt( mu[i] , 1/zsigma^2 , nu )
    mu[i] <- zbeta0 + zbeta1 * zx[i]
  }
  zbeta0 ~ dnorm( 0 , 1/(10)^2 )
  zbeta1 ~ dnorm( 0 , 1/(10)^2 )
  zsigma ~ dunif( 1.0E-3 , 1.0E+3 )
  nu <- nuMinusOne+1
  nuMinusOne ~ dexp(1/29.0)
}
```

Some of the parameter names in the code above are preceded by the letter “z” because the data will be standardized before being sent to the model, as will be described later, and standardized data often are referred to by the letter “z.”

One of the nice features of JAGS (and Stan) is that arguments in distributions are functionally evaluated, so we can put a statement with operations in the place of an argument instead of only a variable name. For example, in the `dt` distribution earlier, the precision argument has the expression `1/zsigma^2` instead of just a variable name. We could do the same thing for `dt`'s other arguments. In particular, instead of assigning a value to `mu[i]` on a separate line, we could simply insert `zbeta0 + zbeta1 * zx[i]` in place of `mu[i]` in the mean argument of `dt`. Thus, assignment arrows in diagrams like

[Figure 17.2](#) do not necessarily need a separate line in the JAGS code. However, when the expression is put directly into an argument, JAGS cannot record the value of the expression throughout the chain. Thus, in the form shown earlier, `mu[i]` is available for recording along with the parameter values at every step. We are not interested in `mu[i]` at every step; therefore, the revised code (exhibited later) will put its expression directly into the mean argument of `dt`. On the other hand, the expression for `nu` will remain on a separate line (as shown above) because we do want an explicit record of its value.

### 17.2.1.1 Standardizing the data for MCMC sampling

In principle, we could run the JAGS code, shown earlier, on the “raw” data. In practice, however, the attempt often fails. There’s nothing wrong with the mathematics or logic; the problem is that credible values of the slope and intercept parameters tend to be strongly correlated, and this narrow diagonal posterior distribution is difficult for some sampling algorithms to explore, resulting in extreme inefficiency in the chains.

You can get an intuition for why this happens by considering the data in the upper panel of [Figure 17.3](#). The superimposed lines are a smattering of credible regression lines from the posterior MCMC chain. All the regression lines must go through the midst of the data, but there is uncertainty in the slope and intercept. As you can see, if the slope is steep, then the  $y$  intercept (i.e., the  $y$  value of the regression line when  $x = 0$ ) is low, at a large negative value. But if the slope is not so steep, then the  $y$  intercept is not so low. The credible slopes and intercepts trade off. The middle-right panel of [Figure 17.3](#) shows the posterior distribution of the jointly credible values of the slope  $\beta_1$  and the intercept  $\beta_0$ . You can see that the correlation of the credible values of the slope and intercept is extremely strong.

MCMC sampling from such a tightly correlated distribution can be difficult. Gibbs sampling gets stuck because it keeps “bumping into the walls” of the distribution. Recall how Gibbs sampling works: One parameter is changed at a time. Suppose we want to change  $\beta_0$  while  $\beta_1$  is constant. There is only a very small range of credible values for  $\beta_0$  when  $\beta_1$  is constant, so  $\beta_0$  does not change much. Then we try changing  $\beta_1$  while  $\beta_0$  is constant. We run into the same problem. Thus, the two parameter values change only very slowly, and the MCMC chain is highly autocorrelated. Other sampling algorithms can suffer the same problem if they are not able to move quickly along the long axis of the diagonal distribution. There is nothing wrong with this behavior in principle, but in practice, it requires us to wait too long before getting a suitably representative sample from the posterior distribution.

There are (at least) two ways to make the sampling more efficient. One way is to change the sampling algorithm itself. For example, instead of using Gibbs sampling (which is one of the main methods in JAGS), we could try Hamiltonian Monte Carlo (HMC) in Stan. We will explore this option in a later section. A second way is to transform the data so that credible regression lines do not suffer such strong correlation between their slopes and intercepts. It is this option we explore presently.

Look again at the data with the superimposed regression lines in [Figure 17.3](#). The problem arises because the arbitrary position of zero on the  $x$  axis is far away from the data. If we simply slid the axis so that zero fell under the mean of the data, then the regression lines could tilt up or down without any big changes on the  $y$  intercept. Sliding the axis so that zero falls under the mean is equivalent to subtracting the mean of the  $x$  values from each  $x$  value. Thus, we can transform the  $x$  values by “*mean centering*” them. This alone would solve the parameter-correlation problem.

But if we’re going to bother to mean center the  $x$  values, we might as well completely standardize the data. This will allow us to set vague priors for the standardized data that will be valid no matter the scale of the raw data. Standardizing simply means re-scaling the data relative to their mean and standard deviation:

$$z_x = \frac{(x - M_x)}{SD_x} \quad \text{and} \quad z_y = \frac{(y - M_y)}{SD_y} \quad (17.1)$$

where  $M_x$  is the mean of the  $x$  values and  $SD_x$  is the standard deviation of the  $x$  values. It is easy to prove, using simple algebra, that the mean of the resulting  $z_x$  values is zero, and the standard deviation of the resulting  $z_x$  values is one, for any initial data set. Thus, standardized values are not only mean centered, but are also stretched or shrunken so that their standard deviation is one.

We will send the standardized data to the JAGS model, and JAGS will return credible parameter values for the standardized data. We then need to convert the parameter values back to the original raw scales. Denote the intercept and slope for standardized data as  $\xi_0$  and  $\xi_1$  (Greek letter “zeta”), and denote the predicted value of  $y$  as  $\hat{y}$ . Then:

$$\begin{aligned} z_{\hat{y}} &= \xi_0 + \xi_1 z_x && \text{by definition of the model} \\ \frac{(\hat{y} - M_y)}{SD_y} &= \xi_0 + \xi_1 \frac{(x - M_x)}{SD_x} && \text{from Equation 17.1} \\ \hat{y} &= \underbrace{\xi_0 SD_y + M_y - \xi_1 M_x SD_y / SD_x}_{\beta_0} + \underbrace{\xi_1 SD_y / SD_x}_{} x && (17.2) \end{aligned}$$

Thus, for every combination of  $\xi_0, \xi_1$  values, there is a corresponding combination of  $\beta_0, \beta_1$  values specified by [Equation 17.2](#). To get from the  $\sigma$  value of the standardized scale to the  $\sigma$  value on the original scale, we simply multiply by  $SD_y$ . The normality parameter remains unchanged because it refers to the relative shape of the distribution and is not affected by the scale parameter.<sup>2</sup>

<sup>2</sup> The fact that  $v$  is unchanged by standardization comes from the formula for the  $t$  distribution, which shows that the standardized score of  $x$  is isolated in its core, with no other role:  $p(x|v, \mu, \sigma) = \frac{\Gamma(\frac{v+1}{2})}{\Gamma(\frac{v}{2}) \sqrt{\pi v} \sigma} \left(1 + \frac{1}{v} \left(\frac{x-\mu}{\sigma}\right)^2\right)^{-\frac{v+1}{2}}$ .

There is one more choice to make before implementing this method: Should we use R to standardize the data, then run JAGS on the standardized data, and then in R transform the parameters back to the original scale? Or, should we send the original data to JAGS, and do the standardization and transformation back to original scale, all in JAGS? Either method can work. We will do the transformation in JAGS. The reason is that this method keeps an integrated record of the standardized and original-scale parameters in JAGS' coda format, so convergence diagnostics can be easily run. This method also gives me the opportunity to show another aspect of JAGS programming that we have not previously encountered, namely, the **data block**.

The data are denoted as vectors  $x$  and  $y$  on their original scale. No other information is sent to JAGS. In the JAGS model specification, we first standardize the data by using a data block, like this:

```
data {
  Ntotal <- length(y)
  xm <- mean(x)
  ym <- mean(y)
  xsd <- sd(x)
  ysd <- sd(y)
  for ( i in 1:length(y) ) {
    zx[i] <- ( x[i] - xm ) / xsd
    zy[i] <- ( y[i] - ym ) / ysd
  }
}
```

All the values computed in the data block can be used in the **model block**. In particular, the model is specified in terms of the standardized data, and the parameters are preceded by a letter  $z$  to indicate that they are for the standardized data. Because the data are standardized, we know that the credible slope cannot greatly exceed the range  $\pm 1$ , so a standard deviation of 10 on the slope makes it extremely flat relative to its possible credible values. Similar considerations apply to the intercept and noise, such that the following model block has extremely vague priors no matter what the original scale of the data:

```
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dt( zbeta0 + zbeta1 * zx[i] , 1/zsigma^2 , nu )
  }
  # Priors vague on standardized scale:
  zbeta0 ~ dnorm( 0 , 1/10^2 )
  zbeta1 ~ dnorm( 0 , 1/10^2 )
  zsigma ~ dunif( 1.0E-3 , 1.0E+3 )
  nu <- nuMinusOne+1
  nuMinusOne ~ dexp(1/29.0)
  # Transform to original scale:
  beta1 <- zbeta1 * ysd / xsd
```

```

beta0 <- zbeta0 * ysd + ym - zbeta1 * xm * ysd / xsd
sigma <- zsigma * ysd
}

```

Notice at the end of the model block, above, the standardized parameters are transformed to the original scale, using a direct implementation of the correspondences in [Equation 17.2](#).

A high-level script for using the JAGS model above is named `Jags-Ymet-Xmet-Mrobust-Example.R`, with functions themselves defined in `Jags-Ymet-Xmet-Mrobust.R`. Running the script produces the graphs shown in [Figures 17.3](#) and [17.4](#). Before discussing the results, let's explore how to implement the model in Stan.

## 17.2.2. Robust linear regression in Stan

Recall from Section 14.1 (p. 400) that Stan uses Hamiltonian dynamics to find proposed positions in parameter space. The trajectories use the gradient of the posterior distribution to move large distances even in narrow distributions. Thus, HMC by itself, without data standardization, should be able to efficiently generate a representative sample from the posterior distribution.

### 17.2.2.1 Constants for vague priors

The only new aspect of the Stan implementation is the setting of the priors. Because the data will not be standardized, we cannot use the constants from the priors in the JAGS model. Instead, we need the constants in the priors to be vague on whatever scale the data happen to fall on. We could query the user to supply a prior, but instead we will let the data themselves suggest what is typical, and we will set the priors to be extremely broad relative to the data (just as we did in the previous chapter).

A regression slope can take on a maximum value of  $SD_y/SD_x$  for data that are perfectly correlated. Therefore, the prior on the slope will be given a standard deviation that is large compared to that maximum. The biggest that an intercept could be, for data that are perfectly correlated, is  $M_x SD_y/SD_x$ . Therefore, the prior on the intercept will have a standard deviation that is large compared to that maximum. The following model specification for Stan implements these ideas:

```

data {
  int<lower=1> Ntotal ;
  real x[Ntotal] ;
  real y[Ntotal] ;
  real meanY ;
  real sdY ;
  real meanX ;
  real sdX ;
}
transformed data { // for constants in priors
  real unifLo ;

```

```

real unifHi ;
real expLambda ;
real beta0sigma ;
real betalsigma ;
unifLo <- sdY/1000 ;
unifHi <- sdY*1000 ;
expLambda <- 1/29.0 ;
betalsigma <- 10*fabs(sdY/sdX) ; // fabs is absolute value
beta0sigma <- 10*fabs(meanX*sdY/sdX) ;
}
parameters {
  real beta0 ;
  real beta1 ;
  real<lower=0> nuMinusOne ;
  real<lower=0> sigma ;
}
transformed parameters {
  real<lower=0> nu ; // actually lower=1
  nu <- nuMinusOne + 1 ;
}
model {
  sigma ~ uniform( unifLo , unifHi ) ;
  nuMinusOne ~ exponential( expLambda ) ;
  beta0 ~ normal( 0 , beta0sigma ) ;
  beta1 ~ normal( 0 , betalsigma ) ;
  for ( i in 1:Ntotal ) {
    y[i] ~ student_t( nu , beta0 + beta1 * x[i] , sigma ) ;
  }
}
}

```

A script for the Stan model above is named Stan-Ymet-Xmet-Mrobust-Example.R, with functions defined in Stan-Ymet-Xmet-Mrobust.R. Running it produces graphs just like those shown in Figures 17.3 and 17.4. Before discussing the results, let's briefly compare the performance of Stan and JAGS.

### 17.2.3. Stan or JAGS?

The Stan and JAGS implementations were both run on the data in Figure 17.4, and they produced very similar results in terms of the details of the posterior distribution. For a comparison, both implementations were run with 500 adaptation steps, 1000 burn-in steps, and 20,000 saved steps with no thinning, using four chains. On my modest desktop computer, a run in JAGS took about 180 seconds, while a run in Stan (including compilation) took about 485 seconds (2.7 times as long). The ESS in JAGS was quite different for different parameters, being about 5000 for  $\nu$  and  $\sigma$  but about 16,000 for  $\beta_0$  and  $\beta_1$ . The effective sample size (ESS) in Stan was more consistent across parameters but not as high as JAGS for some, being about 8000 for  $\nu$  and  $\sigma$  but only about 7000 for  $\beta_0$  and  $\beta_1$ .

Thus, Stan does indeed go a long way toward solving the correlated-parameter problem for which JAGS needed the intervention of standardizing the data. Stan also samples the normality and scale parameters with less autocorrelation than JAGS, as we saw previously in Figure 16.10 (p. 467). HMC deftly navigates through posterior distributions where other samplers may have troubles. But Stan took longer in real time and yielded a smaller ESS for the main parameters of interest, namely the slope and intercept. Stan's efficiency with this model might be improved if we used standardized data as we did for JAGS, but a main point of this demonstration was to show that HMC is able to navigate difficult posterior distributions. Stan's efficiency (and JAGS') might also be improved if the  $t$  distribution were reparameterized, as explained in the Stan Reference Manual. Although subsequent programs in this section focus on JAGS, keep in mind that it is straight forward to translate the programs into Stan.

#### 17.2.4. Interpreting the posterior distribution

Now that we are through all the implementation details, we will discuss the results of the analysis shown in Figures 17.3 and 17.4. Recall that both figures show results from the same data generator, but Figure 17.3 has only 30 data points while Figure 17.4 has 300. By comparing the marginal distributions of the posterior across the two figures, you can see that the modal estimates of the slope, intercept, and scale are about the same, but the certainty of the estimate is much tighter for 300 data points than for 30 data points. For example, with  $N = 30$ , the slope has modal estimate of about 4.5 (pounds per inch), with a 95% HDI that extends from about 2.0 to 7.0. When  $N = 300$ , again the slope has modal estimate of about 4.5, but the 95% HDI is narrower, extending from about 3.6 to 5.2.

It is interesting to note that the normality parameter does not have the same estimate for the two data sets. As the data set gets bigger, the estimate of the normality parameter gradually gets smaller, overcoming the prior which remains fairly dominant with  $N = 30$ . (Recall the discussion of the exponential prior on  $\nu$  in Section 16.2.1, p. 462.) With larger data sets, there is more opportunity for outliers to appear, which are inconsistent with large values of  $\nu$ . (Of course, if the data are genuinely normally distributed, then larger data sets just reinforce large values of the normality parameter.)

In some applications, there is interest in extrapolating or interpolating trends at  $x$  values sparsely represented in the current data. For instance, we might want to predict the weight of a person who is 50 inches tall. A feature of Bayesian analysis is that we get an entire distribution of credible predicted values, not only a point estimate. To get the distribution of predicted values, we step through the MCMC chain, and at each step, we randomly generate simulated data from the model using the parameter values at that step. You can think of this by looking at the vertically plotted noise distributions in Figure 17.4. For any given value of height, the noise distributions show the credible predicted values of weight. By simulating data from all the steps in the

chain, we integrate over all those credible noise distributions. An interesting aspect of the predictive distributions is that they are naturally wider, that is, less certain, the farther away we probe from the actual data. This can be seen most easily in Figure 17.3. As we probe farther from the data, the spread between credible regression lines gets larger, and thus the predicted value gets less certain.

The upper panels of Figures 17.3 and 17.4 show the data with a smattering of credible regression lines and  $t$  distributions superimposed. This provides a visual posterior predictive check, so we can qualitatively sense whether there are systematic deviations of the data from the form implied by the model. For example, we might look for nonlinear trends in the data or asymmetry in the distribution of weights. The data suggest that there might be some positive skew in the distribution of weights relative to the regression lines. If this possibility had great theoretical importance or were strong enough to cast doubt on the interpretation of the parameters, then we would want to formulate a model that incorporated skew in the noise distribution. For more information about posterior predictive checks, see Kruschke (2013b). Curiously, in the present application, the robust linear-regression model does a pretty good job of mimicking the data, even though the data were actually generated by three bivariate normal distributions. Remember, though, that the model only mimics the dependency of  $y$  on  $x$ , and the model does not describe the distribution of  $x$ .

### 17.3. HIERARCHICAL REGRESSION ON INDIVIDUALS WITHIN GROUPS

In the previous applications, the  $j$ th individual contributed a single  $x_j, y_j$  pair. But suppose instead that every individual,  $j$ , contributes multiple observations of  $x_{i|j}, y_{i|j}$  pairs. (The subscript notation  $i|j$  means the  $i$ th observation within the  $j$ th individual.) With these data, we can estimate a regression curve for every individual. If we also assume that the individuals are mutually representative of a common group, then we can estimate group-level parameters too.

One example of this scenario comes from Walker, Gustafson, and Frimer (2007), who measured reading-ability scores of children across several years. Thus, each child contributed several age and reading-score pairs. A regression line can describe each child's reading ability through time, and higher-level distributions can describe the credible intercepts and slopes for the group as a whole. Estimates of each individual are mutually informed by data from all other individuals, by virtue of being linked indirectly through the higher-level distribution. In this case, we might be interested in assessing individual reading ability and rate of improvement, but we might also or instead be interested in estimating the group-level ability and rate of improvement.

As another example, suppose we are interested in how family income varies with the size of the family, for different geographical regions. The U.S. Census Bureau has published data that show the median family income as a function of the number of

persons in a family, for all 50 states, the District of Columbia, and Puerto Rico. In this case, each territory is a “subject” and contributes several pairs of data: Median income for family size 2, median income for family size 3, and so on, up to family size 6 or 7. If we had the actual income for each family, we could use those data in the analysis instead of the median. In this application, we might be interested in estimates for each individual region, or in the overall estimate. The estimated trend in each region is informed by the data from all the other regions indirectly through the higher-level distribution that is informed by all the regions.

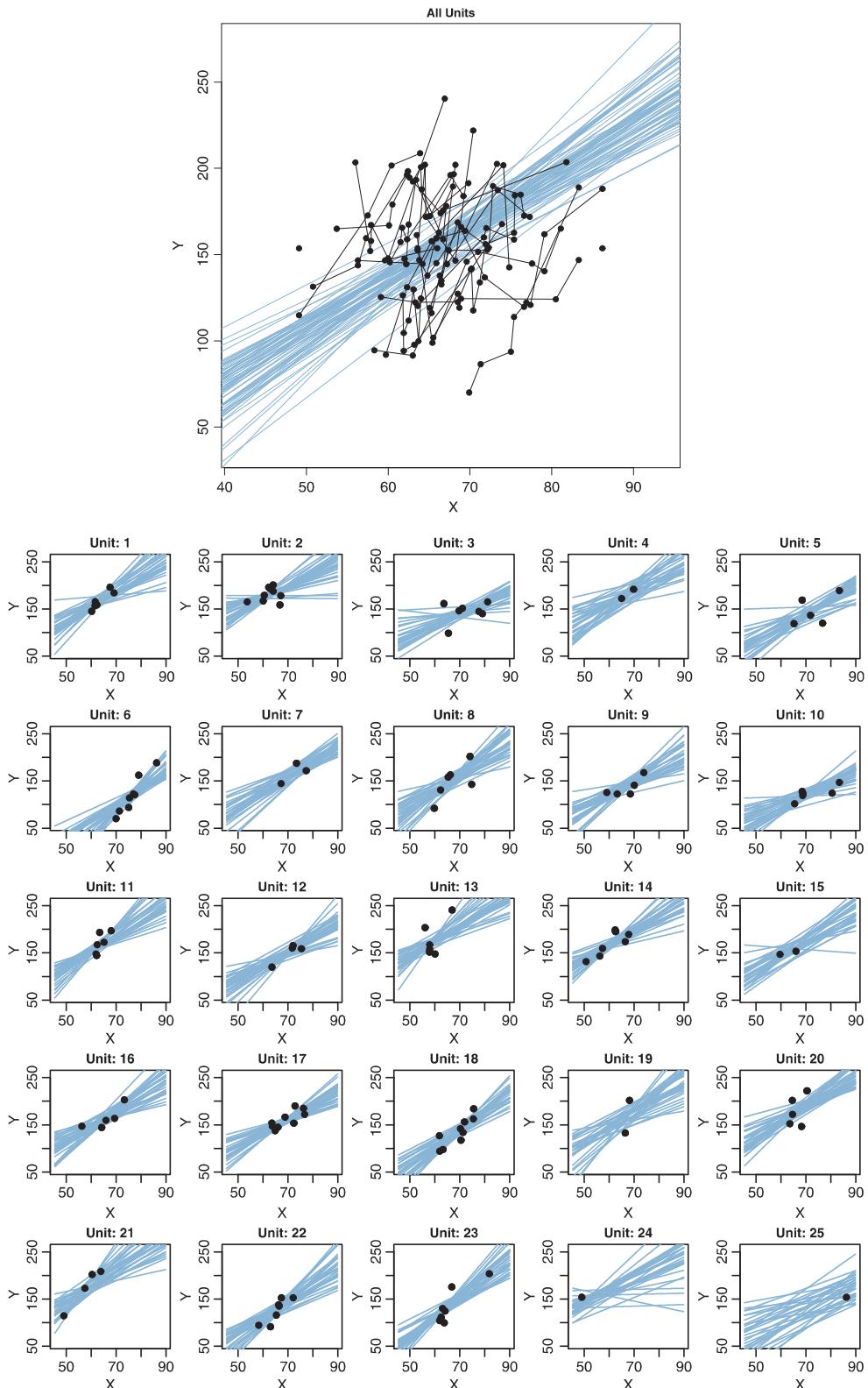
Figure 17.5 shows some fictitious data for a worked example. The scales for  $x$  and  $y$  are roughly comparable to the height and weight example from the previous section, but the data are not intended to be realistic in this case. In these data, each subject contributes several  $x, y$  pairs. You might think of these as growth curves for individual people, but with the caveat that they are not realistic. The top panel of Figure 17.5 shows the data from all the individuals superimposed, with subsets of data from single individuals connected by line segments. The lower panels of Figure 17.5 show the data separated into individuals. Notice that different individuals contribute different numbers of data points, and the  $x$  values for different individuals can be completely different.

Our goal is to describe each individual with a linear regression, and simultaneously to estimate the typical slope and intercept of the group overall. A key assumption for our analysis is that each individual is representative of the group. Therefore, every individual informs the estimate of the group slope and intercept, which in turn inform the estimates of all the individual slopes and intercepts. Thereby we get sharing of information across individuals, and shrinkage of individual estimates toward the overarching mode. We will expand our discussion of shrinkage after twisting our minds around the model specification.

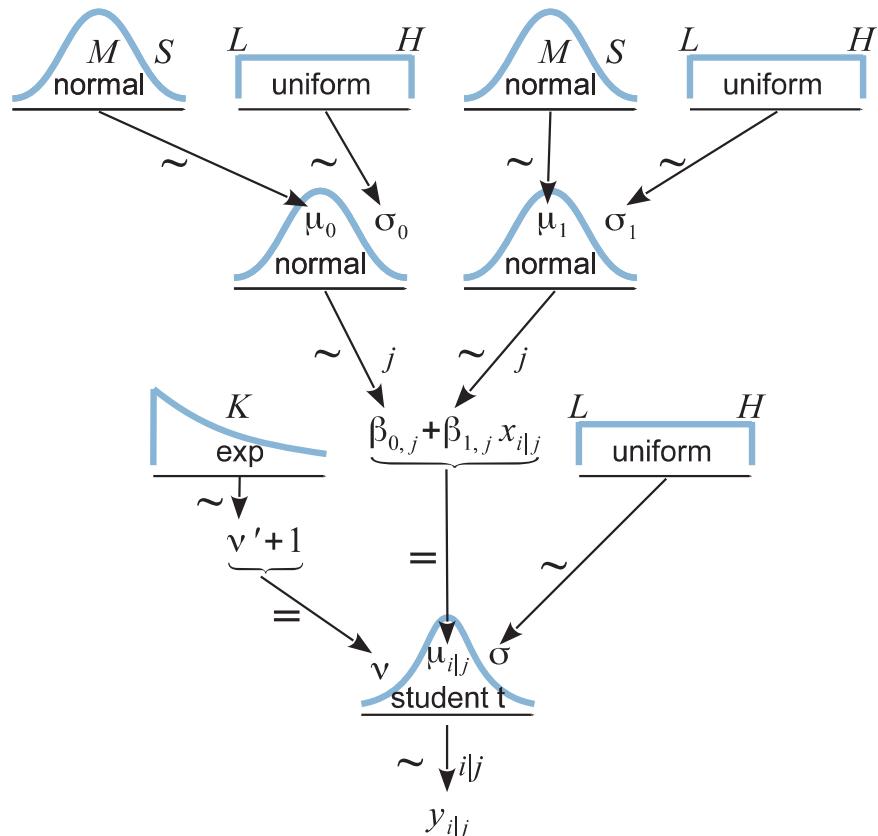
### 17.3.1. The model and implementation in JAGS

Figure 17.6 shows a diagram of the model. It might seem a bit daunting, but if you compare it with Figure 17.2 on p. 480 you will see that it merely adds a level to the basic model we are already very familiar with. The diagram is useful for understanding all the dependencies in the model, and for programming in JAGS (and Stan) because every arrow in the diagram has a corresponding line in the JAGS model specification.

Starting at the bottom of Figure 17.6, we see that the  $i$ th datum within the  $j$ th subject comes from a  $t$  distribution that has a mean of  $\mu_{ij} = \beta_{0,j} + \beta_{1,j}x_{ij}$ . Notice that the intercept and slope are subscripted with  $j$ , meaning that every subject has its own slope and intercept. Moving up the hierarchical diagram, we see that the slopes across the individuals all come from a higher-level normal distribution that has mean denoted  $\mu_1$  and standard deviation of  $\sigma_1$ , both of which are estimated. In other words, the model assumes that  $\beta_{1,j} \sim \text{normal}(\mu_1, \sigma_1)$ , where  $\mu_1$  describes the typical slope of the individuals and  $\sigma_1$  describes the variability of those individual slopes. If every



**Figure 17.5** Fictitious data for demonstrating hierarchical linear regression, with posterior predicted lines superimposed. Upper panel: All data together, with individuals represented by connected segments. Lower panels: Plots of individual data. Notice that the final two subjects have only single data points, yet the hierarchical model has fairly tight estimates of the individual slopes and intercepts.



**Figure 17.6** A model of dependencies for robust hierarchical linear regression. Compare with Figure 17.2 on p. 480.

individual seems to have nearly the same slope, then  $\sigma_1$  is estimated to be small, but if different individuals have very different slopes, then  $\sigma_1$  is estimated to be large. Analogous considerations apply to the intercepts. At the top level of the hierarchy, the group-level parameters are given generic vague priors.

This model assumes that the standard deviation of the noise within each subject is the same for all subjects. In other words, there is a single  $\sigma$  parameter, as shown at the bottom of Figure 17.6. This assumption could be relaxed by providing subjects with distinct noise parameters, all under a higher level distribution. We will see our first example of this approach later, in Figure 19.6 (p. 574).

To understand the JAGS implementation of the model, we must understand the format of the data file. The data format consists of three columns. Each row corresponds to one point of data and specifies the  $x, y$  values of the point and the subject who contributed that point. For the  $i$ th row of the data file, the  $x$  value is  $x[i]$ , the  $y$  value is  $y[i]$ , and the subject index  $j$  is  $s[i]$ . Notice that the index  $i$  in the JAGS model specification is the row of overall data file, not the  $i$ th value within subject  $j$ . The model

specification assumes that the subject indices are consecutive integers, but the data file can use any sort of unique subject identifiers because the identifiers get converted to consecutive integers inside the program. The subject indices will be used to keep track of individual slopes and intercepts.

The JAGS implementation of this model begins by standardizing the data, exactly like the nonhierarchical model of the previous section:

```
data {
  Ntotal <- length(y)
  xm <- mean(x)
  ym <- mean(y)
  xsd <- sd(x)
  ysd <- sd(y)
  for ( i in 1:length(y) ) {
    zx[i] <- ( x[i] - xm ) / xsd
    zy[i] <- ( y[i] - ym ) / ysd
  }
}
```

Then we get to the novel part, where the hierarchical model itself is specified. Notice below that the intercept and slope use nested indexing. Thus, to describe the standardized  $y$  value  $zy[i]$ , the model uses the slope,  $zbeta1[s[i]]$ , of the subject who contributed that  $y$  value. As you read the model block, below, compare each line with the arrows in Figure 17.6:

```
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dt( zbeta0[s[i]] + zbeta1[s[i]] * zx[i], 1/zsigma^2, nu)
  }
  for ( j in 1:Nsubj ) {
    zbeta0[j] ~ dnorm( zbeta0mu , 1/(zbeta0sigma)^2 )
    zbeta1[j] ~ dnorm( zbeta1mu , 1/(zbeta1sigma)^2 )
  }
  # Priors vague on standardized scale:
  zbeta0mu ~ dnorm( 0 , 1/(10)^2 )
  zbeta1mu ~ dnorm( 0 , 1/(10)^2 )
  zsigma ~ dunif( 1.0E-3 , 1.0E+3 )
  zbeta0sigma ~ dunif( 1.0E-3 , 1.0E+3 )
  zbeta1sigma ~ dunif( 1.0E-3 , 1.0E+3 )
  nu <- nuMinusOne+1
  nuMinusOne ~ dexp(1/29.0)
  # Transform to original scale:
  for ( j in 1:Nsubj ) {
    beta1[j] <- zbeta1[j] * ysd / xsd
    beta0[j] <- zbeta0[j] * ysd + ym - zbeta1[j] * xm * ysd / xsd
  }
  betalmu <- zbeta1mu * ysd / xsd
```

```

beta0mu <- zbeta0mu * ysd + ym - zbeta1mu * xm * ysd / xsd
sigma <- zsigma * ysd
}

```

### 17.3.2. The posterior distribution: Shrinkage and prediction

A complete high-level script for running the analysis is in `Jags-Ymet-XmetSsubj-MrobustHier-Example.R`, with the corresponding functions defined in `Jags-Ymet-XmetSsubj-MrobustHier.R`. You will notice that there is nonzero thinning used; this is merely to keep the saved file size relatively small while attaining an ESS of at least 10,000 for relevant parameters. The MCMC file can get large because there are a lot of parameters to store. For example, with 25 subjects, and tracking standardized and original-scale parameters, there are 107 variables recorded from JAGS. The model uses three parallel chains in `runjags` to save time, but it can still take a few minutes. Occasionally, a run will produce a chain that is well behaved for all parameters except the normality parameter  $\nu$ . This chain nevertheless explores reasonable values of  $\nu$  and is well behaved on other parameters. If this is worrisome, simply run JAGS again until all chains are well behaved (or translate to Stan).

Figure 17.5 shows the data with a smattering of credible regression lines superimposed. The lines on the overall data, in the top panel of Figure 17.5, plot the group-level slope and intercept. Notice that the slopes are clearly positive, even though the collective of data points, disconnected from their individual sources, show no obvious upward trend.

The lower panels of Figure 17.5 show individual data with a smattering of credible regression lines superimposed, using individual-level slopes and intercepts. Because of information sharing across individuals, via the higher-level distribution, there is notable shrinkage of the estimates of the individuals. That is to say, the estimates of the individual slopes are pulled together. This shrinkage is especially clear for the final two subjects, who each have only a single data point. Despite this dearth of data, the estimates of the slope and intercept for these subjects are surprisingly tightly constrained, as shown by the fact that the smattering of credible lines is a fairly tight bundle that looks like a blurry version of the posterior predictive lines in the other subjects. Notice that the estimated intercepts of the two final subjects are pulled toward the data values, so that predictions are different for the different individuals. This is another illustration of shrinkage of estimation in hierarchical models, which was introduced in Section 9.3 (p. 245).

## 17.4. QUADRATIC TREND AND WEIGHTED DATA

The U.S. Census Bureau publishes information from the American Community Survey and Puerto Rico Community Survey, including data about family income and

family size.<sup>3</sup> Suppose we are interested in the median family income for different family sizes, in each of the 50 states, Puerto Rico, and the District of Columbia. The data are displayed in Figure 17.7. You can see that the incomes appear to have a nonlinear, inverted-U trend as family size increases. Indeed, if you run the data through the linear-regression model of the previous section, you will notice obvious systematic deviations away from the posterior predicted lines.

Because a linear trend appears to be an inadequate description of the data, we will extend the model to include a quadratic trend. This is not the only way to express a curve mathematically, but it is easy and conventional. A quadratic has the form  $y = b_0 + b_1x + b_2x^2$ . When  $b_2$  is zero, the form reduces to a line. Therefore, this extended model can produce any fit that the linear model can. When  $b_2$  is positive, a plot of the curve is a parabola that opens upward. When  $b_2$  is negative, the curve is a parabola that opens downward. We have no reason to think that the curvature in the family-income data is exactly a parabola, but the quadratic trend might describe the data much better than a line alone. If the posterior distribution on the parameters indicates that credible values of  $b_2$  are far less than zero, we have evidence that the linear model is not adequate (because the linear model would have  $b_2 = 0$ ).

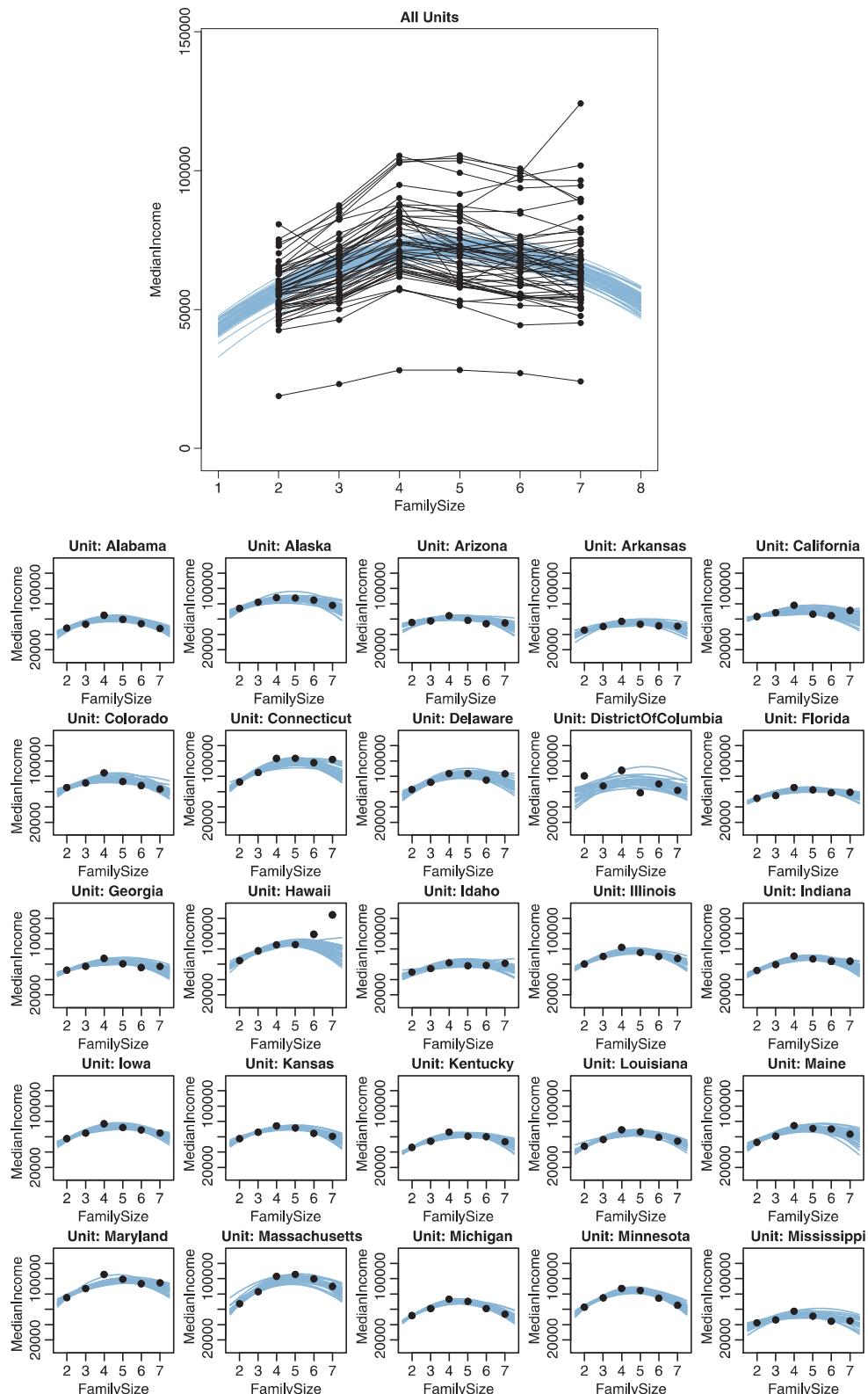
We will expand the model to include a quadratic component for every individual (i.e., state) and for the group level. To make the MCMC more efficient, we will standardize the data and then transform the parameters back to the original scale, just as we did for the linear model (see Equation 17.2, p. 485). Because of the quadratic component, the transformation involves a little more algebra:

$$\begin{aligned} z_{\hat{y}} &= \zeta_0 + \zeta_1 z_x + \zeta_2 z_x^2 && \text{by definition of the model} \\ \frac{(\hat{y} - M_y)}{SD_y} &= \zeta_0 + \zeta_1 \frac{(x - M_x)}{SD_x} + \zeta_2 \frac{(x - M_x)^2}{SD_x^2} && \text{from Equation 17.1} \\ \hat{y} &= \underbrace{\zeta_0 SD_y + M_y - \zeta_1 M_x SD_y / SD_x + \zeta_2 M_x^2 SD_y / SD_x^2}_{\beta_0} \\ &\quad + \underbrace{(\zeta_1 SD_y / SD_x - 2\zeta_2 M_x SD_y / SD_x^2)}_{\beta_1} x + \underbrace{\zeta_2 SD_y / SD_x^2}_{\beta_2} x^2 \end{aligned} \quad (17.3)$$

The correspondences in Equation 17.3 are implemented in JAGS, so JAGS keeps track of the standardized coefficients and the original-scale coefficients.

There is one other modification we will make for the family-income data. The data report the median income at each family size, but the median is based on different numbers of families at each size. Therefore, every median has a different amount of

<sup>3</sup> Data are from [http://www.census.gov/hhes/www/income/data/Fam\\_Inc\\_SizeofFam1.xls](http://www.census.gov/hhes/www/income/data/Fam_Inc_SizeofFam1.xls), retrieved December 11, 2013. Median family income for years 2009–2011.



**Figure 17.7** Upper panel: Median family income for various family sizes, with separate line for each of 50 states and the District of Columbia and Puerto Rico. Lower panels show data of a subset of individual states. Credible quadratic trends are superimposed.

sampling noise. Typically, there are fewer large-sized families than small-sized families, and therefore, the medians for the large-size families are noisier than the medians for the small-size families. The model should take this into account, with parameter estimates being more tightly constrained by the less noisy data points.

Fortunately, the U.S. Census Bureau reported each median with a “margin of error” that reflects the standard error of the data collected for that point. We will use the margin of error to modulate the noise parameter in the model. If the margin of error is high, then the noise parameter should be increased proportionally. If the margin of error is small, then the noise parameter should be decreased proportionally. More formally, instead of using  $y_i \sim \text{normal}(\mu_i, \sigma)$  with the same  $\sigma$  for every datum, we use  $y_i \sim \text{normal}(\mu_i, w_i\sigma)$ , where  $w_i$  reflects the relative standard error for the  $i$ th datum.<sup>4</sup> When  $w_i$  is small, then  $\mu_i$  has relatively little wiggle room around its best fitting value. When  $w_i$  is large, then  $\mu_i$  can deviate farther from its best fitting value without greatly changing the likelihood of the datum. Overall, then, this formulation finds parameter values that better match the data points with small margin of error than the data points with large margin of error. In the JAGS specification below, the original-scale standard errors are divided by their mean, so that a weight of 1.0 implies the mean standard error.

The quadratic component and weighted noise are implemented in the JAGS model as follows:

```
# Standardize the data:
data {
  Ntotal <- length(y)
  xm <- mean(x)
  ym <- mean(y)
  wm <- mean(w)  # standard error of y
  xsd <- sd(x)
  ysd <- sd(y)
  for ( i in 1:length(y) ) {
    zx[i] <- ( x[i] - xm ) / xsd
    zy[i] <- ( y[i] - ym ) / ysd
    zw[i] <- w[i] / wm  # set noise weights relative to mean noise
  }
}
# Specify the model for standardized data:
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dt( zbeta0[s[i]]+zbeta1[s[i]]*zx[i]+zbeta2[s[i]]*zx[i]^2 ,
      1/(zw[i]*zsigma)^2 , nu )
```

<sup>4</sup> In this formulation, the weights  $w_i$  represent noise or error. In other formulations of weighted regression, the weights represent precision or inverse error variance, and therefore,  $\sigma$  is divided by the weight instead of multiplied.

```

}
for ( j in 1:Nsubj ) {
  zbeta0[j] ~ dnorm( zbeta0mu , 1/(zbeta0sigma)^2 )
  zbeta1[j] ~ dnorm( zbeta1mu , 1/(zbeta1sigma)^2 )
  zbeta2[j] ~ dnorm( zbeta2mu , 1/(zbeta2sigma)^2 )
}
# Priors vague on standardized scale:
zbeta0mu ~ dnorm( 0 , 1/(10)^2 )
zbeta1mu ~ dnorm( 0 , 1/(10)^2 )
zbeta2mu ~ dnorm( 0 , 1/(10)^2 )
zsigma ~ dunif( 1.0E-3 , 1.0E+3 )
zbeta0sigma ~ dunif( 1.0E-3 , 1.0E+3 )
zbeta1sigma ~ dunif( 1.0E-3 , 1.0E+3 )
zbeta2sigma ~ dunif( 1.0E-3 , 1.0E+3 )
nu <- nuMinusOne+1
nuMinusOne ~ dexp(1/29.0)
# Transform to original scale:
for ( j in 1:Nsubj ) {
  beta2[j] <- zbeta2[j]*ysd/xsd^2
  beta1[j] <- zbeta1[j]*ysd/xsd - 2*zbeta2[j]*xm*ysd/xsd^2
  beta0[j] <- zbeta0[j]*ysd + ym - zbeta1[j]*xm*ysd/xsd + zbeta2[j]*xm^2*ysd/xsd^2
}
beta2mu <- zbeta2mu*ysd/xsd^2
beta1mu <- zbeta1mu*ysd/xsd - 2*zbeta2mu*xm*ysd/xsd^2
beta0mu <- zbeta0mu*ysd + ym - zbeta1mu*xm*ysd/xsd + zbeta2mu*xm^2*ysd/xsd^2
sigma <- zsigma * ysd
}

```

In the JAGS specification above, notice that the mean argument of the student *t* distribution includes the new quadratic component. And notice that the precision argument of the student *t* distribution multiplies *zsigma* by the datum-specific noise weight, *zw[i]*. After the likelihood function, the remaining additions to the model specification merely specify the prior on the quadratic terms and implement the new transformation to the original scale (Equation 17.3). The complete script and functions for the model are in *Jags-Ymet-XmetSsubj-MrobustHierQuadWt-Example.R* and *Jags-Ymet-XmetSsubj-MrobustHierQuadWt.R*. The script is set up such that if you have a data file without standard errors for each data point, the weight name is simply omitted (or explicitly set as *wName=NULL*).

The Stan version of the specification above is provided in [Exercise 17.3](#).

### 17.4.1. Results and interpretation

As you can see from the superimposed credible regression curves in [Figure 17.7](#), the curvature is quite prominent. In fact, the 95% HDI on the quadratic coefficient (not shown here, but displayed by the script) goes from  $-2200$  to  $-1700$ , which is far from zero. Thus, there is no doubt that there is a nonlinear trend in these data.

Interestingly, shrinkage from the hierarchical structure informs the estimates for each state. For example, inspect the subpanel for Hawaii in Figure 17.7. Its credible trend lines are curved downward even though its data in isolation curve upward. Because the vast majority of states show downward curvature in their data, and because Hawaii is assumed to be like the other states, the most credible estimates for Hawaii have a downward curvature. If Hawaii happened to have had a ginormous amount of data, so that its noise weights were extremely small, then its individual credible curves would more closely match its individual trend. In other words, shrinkage from the group level is a compromise with data from the individual.

Care must be taken when interpreting the linear component, that is, the slope. The slope is only meaningful in the context of adding the quadratic component. In the present application, the 95% HDI on the slope goes from 16,600 to 21,800. But this slope, by itself, greatly overshoots the data because the quadratic component subtracts a large value.

The weighting of data by their standard error is revealed by greater uncertainty at large family sizes than small. For example, consider the subpanel for California (or almost every other state) in Figure 17.7. The credible curves have a narrow spread at family size 2 but large spread at family size 7. This is caused in part by the fact that most of the data for large family sizes have large standard errors, and therefore, the parameters have more “slop” or flexibility at the large family sizes. (Another cause of spread in credible curves is variability within and between units.)

Finally, the posterior distribution reveals that these data have outliers within individual states because the normality parameter is estimated to be very small. Almost all of the posterior distribution is below  $\nu = 4$ . This suggests that most of the data points within each individual state fall fairly close to a quadratic-trend line, and the remaining points are accommodated by large tails of the  $t$  distribution.

### 17.4.2. Further extensions

Even this model has many simplifications for the purpose of clarity. One simplification was the use of only linear and quadratic trends. It is easy in Bayesian software to include any sort of non-linear trend, such as higher-order polynomial terms, or sinusoidal trends, or exponential trends or any other mathematically defined trend.

This model assumed a single underlying noise for all individuals. The noise could be modulated by the relative standard error of each datum, but the reference quantity,  $\sigma$ , was the same for all individuals. We do not need to make this assumption. Different individuals might have different inherent amounts of noise in their data. For example, suppose we were measuring a person’s blood pressure at various times of day. Some people might have relatively consistent blood pressure while others might have greatly varying blood pressure. When there is sufficient data for each individual, it could be useful to expand the model to include subject-specific standard deviations or “noise”

parameters. This lets the less noisy individuals have a stronger influence on the group-level estimate than the noisier individuals. The individual noise parameters could be modeled as coming from a group-level gamma distribution:  $\sigma_i \sim \text{gamma}(r, s)$ , where  $r$  and  $s$  are estimated, so that the gamma distribution describes the variation of the noise parameter across individuals. We will see our first example of this approach later in Figure 19.6 (p. 574).

Another simplification was assuming that intercepts, slopes, and curvatures are normally distributed across individuals. But it could be that there are outliers: Some individuals might have very unusual intercepts, slopes, or curvatures. Therefore, we could use  $t$  distributions at the group level. It is straight forward in Bayesian software to change the distributions from normal to student  $t$ . The model thereby becomes robust to outliers at multiple levels. But remember that estimating the normality parameter relies on data in the tails, so if the data set has few individuals and no extreme individuals, there will be nothing gained by using  $t$  distributions at the group level.

Another simplification in the model was that it had no explicit parameters to describe covariation of the intercepts, slopes, and curvatures across individuals. For example, rat pups that are born relatively small tend to grow at a smaller rate than rat pups that are born relatively larger: The intercept (weight at birth) and slope (growth rate) naturally covary. In other applications, the correlation could be the opposite, of course. The point is that the present model has no explicit parameter to capture such a correlation. It is straight forward in Bayesian software to use a multivariate normal prior on the intercept, slope, and curvature parameters. The multivariate normal has explicit covariance parameters, which are estimated along with the other parameters. (See, e.g., the “birats” example from WinBUGS Examples Volume 2 at [http://www.mrc-bsu.cam.ac.uk/wp-content/uploads/WinBUGS\\_Vol2.pdf](http://www.mrc-bsu.cam.ac.uk/wp-content/uploads/WinBUGS_Vol2.pdf).)

## 17.5. PROCEDURE AND PERILS FOR EXPANDING A MODEL

### 17.5.1. Posterior predictive check

Comparing the data to the posterior predictions of the model is called a *posterior predictive check*. When there appear to be systematic discrepancies that would be meaningful to address, you should consider expanding or changing the model so it may be a better description of the data.

But what extension or alternative model should you try? There is no uniquely correct answer. The model should be both meaningful and computationally tractable. Meaning can come from mere familiarity in the catalog of functions learned in traditional math courses. This familiarity and simplicity was what motivated the use of a quadratic trend for the family-income data. Alternatively, meaning can come from theories of the application domain. For example, for family income, we might imagine that total income rises as the square root of the number of adults in the family (because each adult

has the potential to bring in money to the family) and declines exponentially with the number of children in the family (because each child requires more time away from income generation by the adults). That model is completely fictitious and is mentioned merely for illustration.

The extended model is intended to better describe the data. One way to extend a model is to leave the original model “nested” inside the extended model, such that all the original parameters and mathematical form can be recovered from the extended model by setting the extended parameters to specific constants, such as zero. An example is extending the linear model to a quadratic-trend model; when the curvature is set to zero, the nested linear model is recovered. In the case of nested models, it is easy to check whether the extension fits the data better merely by inspecting the posterior distribution on the new parameters. If the new parameter values are far from the setting that produces the original model, then we know that the new parameters are useful for describing the data.

Instead of adding new parameters to a nested existing model, we might be inspired to try a completely different model form. To assess whether the new model describes the data better than the original model, Bayesian model comparison could be used, in principle. The priors on the parameters within the two models would need to be equivalently informed, as was discussed in Section 10.6.1 (p. 294).

Some people caution that looking at the data to inspire a model form is “double dipping” into the data, in the sense that the data are being used to change the prior distribution on the model space. Of course, we can consider any model space we care to, and the motivation for that model space can evolve through time and come from many different sources, such as different theories and different background literature. If a set of data suggests a novel trend or functional form, presumably the prior probability on that novel trend was small unless the analyst realizes the same trend had occurred unheralded in previous results. The analyst should keep the low prior probability in mind, either formally or informally. Moreover, novel trends are retrospective descriptions of particular data sets, and the trends need to be confirmed by subsequent, independently collected data.

Another approach to a posterior predictive check is to create a posterior predictive sampling distribution of a measure of discrepancy between the predictions and the data. In this approach, a “Bayesian  $p$  value” indicates the probability of getting the data’s discrepancy, or something more extreme, from the model. If the Bayesian  $p$  value is too small, then the model is rejected. From my discussion of  $p$  values in Chapter 11, you might imagine that I find this approach to be problematic. And you would be right. I discuss this issue in more detail in an article (Kruschke, 2013b).

### 17.5.2. Steps to extend a JAGS or Stan model

One of the great benefits of Bayesian software is its tremendous flexibility for specifying models that are theoretically meaningful and appropriate to the data. To take advantage of this flexibility, you will want to be able to modify existing programs. An example of

such a modification was presented in this chapter in the extension from the linear-trend model to the quadratic-trend model. The steps in such a modification are routine, as outlined here:

- Carefully specify the model with its new parameters. It can help to sketch a diagram like Figure 17.6 to make sure that you really understand all the parameters and their priors. The diagram can also help coding the model, because there is usually a correspondence of arrows to lines of code. For example when extending a linear model to a quadratic, the diagram gets just one more branch, for the quadratic coefficient, that is structurally identical to the branch for the linear coefficient. The expression for the mean must be extended with the quadratic term:  $\mu_{ij} = \dots + \beta_{2,j} x_{ij}^2$ , using notation in the code that is consistent with the already-established code. The group-level distribution over the unit-level quadratic coefficients must also be included. Again, this is easy because the model structure of the quadratic coefficient is exactly analogous to the model structure of the linear coefficient.
- Be sure all the new parameters have sensible priors. Be sure the priors for the old parameters still make sense in the new model.
- If you are defining your own initial values for the chains, define initial values for all the new parameters, and make sure that the initial values of the old parameters still make sense for the extended model. JAGS will initialize any stochastic variables that are not given explicit initial values. It is often easiest to let JAGS initialize parameters automatically.
- Tell JAGS to track the new parameters. JAGS records only those parameters that you explicitly tell it to track. Stan tracks all nonlocal variables (such as parameters and transformed parameters) by default.
- Modify the summary and graphics output to properly display the extended model. I have found that this step is the most time-consuming and often the most error-prone. Because the graphics are displayed by R, not by JAGS, you must modify all of your R graphics code to be consistent with the modified model in JAGS. Depending on the role of the parameter in the model and its meaning for interpreting the data, you might want a plot of the parameter's marginal posterior distribution, and perhaps pairs plots of the parameter crossed with other parameters to see if the parameters are correlated in the posterior, and probably specialized displays of the posterior predictive that show trends and predicted spread of data superimposed on the data, and possibly differences between parameters or other comparisons and contrasts, etc.

### 17.5.3. Perils of adding parameters

When including new parameters, the extended model has new flexibility in fitting the data. There may be a far wider range of credible values for the previous parameters. For example, consider again the data that were introduced for the hierarchical linear model, back in Figure 17.5 (p. 492). We can easily use the hierarchical quadratic-trend model

on these data, which yields the posterior credible trends shown in Figure 17.8.<sup>5</sup> Notice that there are many positive or negative curvatures that are consistent with the data. The curvature and slope trade-off strongly in their fit to the original-scale data, such that a positive curvature with small slope fits the data as well as a negative curvature with large slope. Thus, the introduction of the curvature parameter makes the slope parameter less certain, even though the posterior distribution of the curvature parameter is centered nearly at zero. Specifically, in the linear-trend fit, wherein the curvature is implicitly forced to be exactly zero, the 95% HDI on the slope goes from +2.2 to +4.1 (not shown here, but produced in the output of the script). In the quadratic-trend fit, the curvature has a mode very near zero with a 95% HDI from about -0.10 to +0.07, but the 95% HDI of the slope goes from about -7.0 to +16.5. In other words, even though the quadratic-trend model indicates that there is little if any quadratic trend in the data, the estimate of the slope is blurred. The magnitude of this apparent blurring of the slope depends strongly on the data scale, however. Using standardized data, the 95% HDI on the slope parameter changes only slightly.

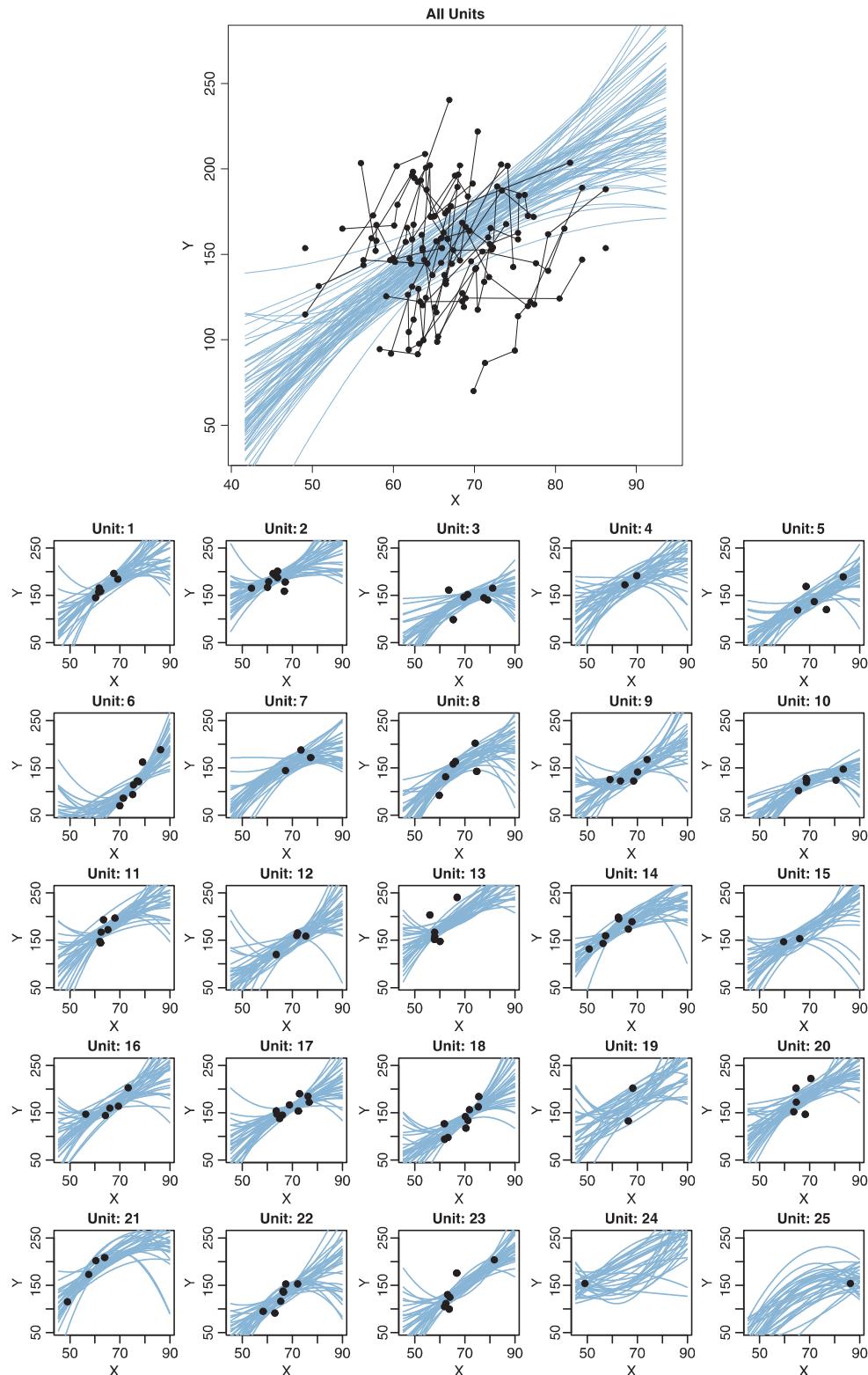
This increase in uncertainty of a parameter estimate is a form of penalizing complexity: For nested models, in which the simpler model is the more complex model with one of the complex model's parameters fixed (e.g., to zero), then the estimates of the nested parameters will tend to be less precise in the more complex model. The degree to which a parameter estimate broadens, when expanding the model, depends on the model structure, the data, and the parameterization. The "blurring" can be especially pronounced if the parameter values can trade-off and fit the data equally well.

A different way of penalizing complexity comes from Bayesian model comparison. The prior on the larger parameter space dilutes the prior probability of any particular combination of parameter values, thereby favoring the simpler model unless the data are much better fit by a combination of parameter values that is not available to the simpler model. It is worth recapitulating that in Bayesian model comparison, the priors for the two models would need to be equivalently informed, as was discussed in Section 10.6.1 (p. 294).

## 17.6. EXERCISES

Look for more exercises at <https://sites.google.com/site/doingbayesiandataanalysis/>

<sup>5</sup> In JAGS, many runs take a long burn-in to converge, and even after that, most runs yield highly auto-correlated chains for the normality parameter. But all chains show similar smooth marginal distributions on the normality parameter, and all chains are well behaved on the other parameters. In Stan, burn-in is rapid, and the normality parameter is smoothly sampled, but most runs yield chains that get temporarily stuck and show unrepresentative bumps in the group-level means. Exercise 17.3 has you try this yourself.



**Figure 17.8** Fitting the data of [Figure 17.5](#) with the quadratic-trend model. The flexibility of the quadratic trend yields more uncertainty in the linear trend, despite the fact that the modal estimate of the quadratic trend is nearly zero.

**Exercise 17.1. [Purpose: Apply linear model to quadratic data and do a posterior predictive check.]**

(A) Change the script Jags-Ymet-XmetSsubj-MrobustHier-Example.R so it uses the family income data. (Notice how simple it is to do that.) Are the data well described by the linear model? What exactly is the “systematic discrepancy” between model and data? Don’t just say, “the data are curved.” What exactly does that mean in terms of how the data deviate from the model and where?

(B) Is there a way of rejecting the linear model without reference to any other model? In other words, how might you compute a “Bayesian  $p$  value” for this situation? Should you? See Kruschke (2013b) for information and references.

**Exercise 17.2. [Purpose: Observe the autocorrelation in JAGS when data are not standardized.]** Change the (simple, nonhierarchical) JAGS program Jags-Ymet-Xmet-Mrobust.R so that it uses the raw data instead of the standardized data. Be sure to rename it so that you don’t destroy the original program. You will have to remove the standardization from the data block and remove the transformation to the original scale. Make sure that the prior is appropriate for the original scale (perhaps use the prior in the Stan version for guidance). Show the diagnostic graphs of the chains and discuss. Finally, run the chains a long time, with thinning if needed to conserve computer memory, and show that the chains eventually converge to the same posterior as when using standardized data.

**Exercise 17.3. [Purpose: Examining chain convergence in JAGS and Stan.]** As mentioned in Footnote 5 (p. 504), both JAGS and Stan show some difficulty converging when the quadratic-trend model is applied to the fictitious data of Figure 17.5. The JAGS programs are provided in files Jags-Ymet-XmetSsubj-MrobustHierQuadWt-Example.R and Jags-Ymet-XmetSsubj-MrobustHierQuadWt.R. The corresponding Stan programs are provided in files Stan-Ymet-XmetSsubj-MrobustHierQuadWt-Example.R and Stan-Ymet-XmetSsubj-Mrobust HierQuadWt.R. The Stan model specification is shown below so that you can study it and compare it to the JAGS version in Section 17.4 without having to be at your computer:

```
data {
  int<lower=1> Nsubj ;
  int<lower=1> Ntotal ;
  real y[Ntotal] ;
  real x[Ntotal] ;
  real<lower=0> w[Ntotal] ; weight for standard error
  int<lower=1> s[Ntotal] ; subject j
}
transformed data {
  // Standardize the data:
  real zx[Ntotal] ;
  real zy[Ntotal] ;
```

```

real zw[Ntotal] ;
real wm ;
real xm ;
real ym ;
real xsd ;
real ysd ;
xm <- mean(x) ;
ym <- mean(y) ;
wm <- mean(w) ;
xsd <- sd(x) ;
ysd <- sd(y) ;
for ( i in 1:Ntotal ) { // could be vectorized...?
  zx[i] <- ( x[i] - xm ) / xsd ;
  zy[i] <- ( y[i] - ym ) / ysd ;
  zw[i] <- w[i] / wm ;
}
parameters {
  real zbeta0[Nsubj] ;
  real zbeta1[Nsubj] ;
  real zbeta2[Nsubj] ;
  real<lower=0> zsigma ;
  real zbeta0mu ;
  real zbeta1mu ;
  real zbeta2mu ;
  real<lower=0> zbeta0sigma ;
  real<lower=0> zbeta1sigma ;
  real<lower=0> zbeta2sigma ;
  real<lower=0> nuMinusOne ;
}
transformed parameters {
  real<lower=0> nu ;
  real beta0[Nsubj] ;
  real beta1[Nsubj] ;
  real beta2[Nsubj] ;
  real<lower=0> sigma ;
  real beta0mu ;
  real beta1mu ;
  real beta2mu ;
  nu <- nuMinusOne+1 ;
  // Transform to original scale:
  for ( j in 1:Nsubj ) { // could be vectorized...?
    beta2[j] <- zbeta2[j]*ysd/square(xsd) ;
    beta1[j] <- zbeta1[j]*ysd/xsd - 2*zbeta2[j]*xm*ysd/square(xsd) ;
    beta0[j] <- zbeta0[j]*ysd + ym - zbeta1[j]*xm*ysd/xsd
      + zbeta2[j]*square(xm)*ysd/square(xsd) ;
  }
  beta2mu <- zbeta2mu*ysd/square(xsd) ;
  betalmu <- zbeta1mu*ysd/xsd - 2*zbeta2mu*xm*ysd/square(xsd) ;
}

```

```

beta0mu <- zbeta0mu*ysd + ym - zbeta1mu*xm*ysd/xsd
          + zbeta2mu*square(xm)*ysd/square(xsd) ;
sigma <- zsigma * ysd ;
}
model {
zbeta0mu ~ normal( 0 , 10 ) ;
zbeta1mu ~ normal( 0 , 10 ) ;
zbeta2mu ~ normal( 0 , 10 ) ;
zsigma ~ uniform( 1.0E-3 , 1.0E+3 ) ;
zbeta0sigma ~ uniform( 1.0E-3 , 1.0E+3 ) ;
zbeta1sigma ~ uniform( 1.0E-3 , 1.0E+3 ) ;
zbeta2sigma ~ uniform( 1.0E-3 , 1.0E+3 ) ;
nuMinusOne ~ exponential(1/29.0) ;
zbeta0 ~ normal( zbeta0mu , zbeta0sigma ) ; // vectorized
zbeta1 ~ normal( zbeta1mu , zbeta1sigma ) ; // vectorized
zbeta2 ~ normal( zbeta2mu , zbeta2sigma ) ; // vectorized
for ( i in 1:Ntotal ) {
zy[i] ~ student_t(
nu ,
zbeta0[s[i]] + zbeta1[s[i]] * zx[i] + zbeta2[s[i]] * square(zx[i]) ,
zw[i]*zsigma ) ;
}
}

```

Review Section 14.4 (p. 414) for hints about programming Stan.

**(A)** Run Stan on the family-income data, so it achieves the same ESS as the JAGS program for the group-level trend coefficients. How long (in real time) do Stan and JAGS take? Does Stan more consistently converge than JAGS? Does Stan produce better chains for the normality and noise parameters?

**(B)** Now repeat on the fictitious data of Figure 17.8, which typically gives JAGS and Stan troubles of differing sorts as described Footnote 5 (p. 504). Try to produce examples of these troubles and discuss. Which type of trouble is more tolerable, autocorrelation in the normality parameter (in JAGS) or “bumps” in the regression coefficients (in Stan)?